

Information Systems Solutions:

A Project Approach

Information Systems Solutions:

A Project Approach

Richard L. Van Horn

Albert B. Schwarzkopf

R. Leon Price

All of the University of Oklahoma



Boston Burr Ridge, IL Dubuque, IA Madison, WI New York
San Francisco St. Louis Bangkok Bogotá Caracas Kuala Lumpur
Lisbon London Madrid Mexico City Milan Montreal New Delhi
Santiago Seoul Singapore Sydney Taipei Toronto

The McGraw-Hill Companies



INFORMATION SYSTEMS SOLUTIONS: A PROJECT APPROACH

Published by McGraw-Hill/Irwin, a business unit of The McGraw-Hill Companies, Inc., 1221 Avenue of the Americas, New York, NY, 10020. Copyright © 2006 by The McGraw-Hill Companies, Inc. All rights reserved. No part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written consent of The McGraw-Hill Companies, Inc., including, but not limited to, in any network or other electronic storage or transmission, or broadcast for distance learning.

Some ancillaries, including electronic and print components, may not be available to customers outside the United States.

This book is printed on acid-free paper.

1 2 3 4 5 6 7 8 9 0 DOC/DOC 0 9 8 7 6 5

ISBN 0-07-352436-0

Editorial director: Brent Gordon

Executive editor: Paul Ducham

Editorial assistant: Liz Farina

Senior marketing manager: Douglas Reiner

Media producer: Greg Bates

Project manager: Marlena Pechan

Lead production supervisor: Michael R. McCormick

Design coordinator: Cara David

Media project manager: Matthew Perry

Developer, Media technology: Brian Nacik

Typeface: 10/12 Palatino

Compositor: GTS—New Delhi, India Campus

Printer: R. R. Donnelley

Library of Congress Cataloging-in-Publication Data

Van Horn, Richard L.

Information systems solutions : a project approach / Richard L. Van Horn, Albert B. Schwarzkopf, R. Leon Price.

p. cm.

Includes bibliographical references and index.

ISBN 0-07-352436-0 (alk. paper)

1. Information technology. 2. Management information systems. I. Schwarzkopf, Albert B. II. Price, R. Leon. III. Title.

T58.5.V37 2006

658.4'032—dc22

www.mhhe.com

2005053443

About the Authors

Dr. Richard L. Van Horn is President Emeritus of the University of Oklahoma (OU), Regent's Professor of Management Information Systems, and Clarence E. Page Professor of Aviation/Aerospace Studies. He also served at OU as the first director for the Management Information Systems Division and the Center for MIS Studies, an industry-university consortium. Before coming to OU, he served as president of the University of Houston and as a faculty member, associate dean of the Graduate School of Industrial Administration and provost at Carnegie Mellon University. He worked on management applications of computers at the Rand Corporation as a researcher, head of the Management Systems Group, and codirector of the Management Systems Laboratory.

Dr. Van Horn holds a Ph.D. in system sciences from Carnegie-Mellon University, an M.S. degree in industrial management from the Massachusetts Institute of Technology and a B.S. degree in industrial administration with highest honors from Yale University. He received an honorary Doctor of Business from Reitsumeikan University. Dr. Van Horn has coauthored three books: *Automatic Data-Processing Systems*, *Business Data Processing and Programming*, and *Management Information Systems: Progress and Perspectives*. His information technology-related professional activities over the years include national council member, the Institute of Management Sciences; Information Systems department editor, *Management Science*; national lecturer, the Association of Computing Machinery; chairman, Harvard University Visiting Committee for Information Technology; chairman of the board, the National Center for Higher Education Management Systems; and vice chairman of the board, EDUCOM.

Professor Albert B. Schwarzkopf received his Ph.D. in mathematics in 1968 from the University of Virginia and spent 15 years in the mathematics department at the University of Oklahoma. In 1984, he transferred to the OU College of Business Administration to help establish the MIS program. Other OU positions he has held include director of the Division of Management, director of the Statistics Consulting Laboratory, statistician for the Information Systems Program Office and, most recently, director of the Telecomputing Degree Program. He has been heavily involved in the development and teaching of the MIS curriculum and has coordinated the Field Project Course for the last seven years.

Professor Schwarzkopf is a former chair of OU's Faculty Senate and of the University Computing Committee. He twice was selected as the outstanding business faculty member by the student government. His research interests include decision support and end-user computing in MIS and systems modeling in production operations management and general business. He has obtained several grants for research in various facets of MIS and business studies. His articles have been published in *Research Policy*, *International Journal of Production Research*, *Journal of Business Logistics*, *Journal of Retailing and Transportation Research*, and *Communications of the ACM*, among others.

Dr. R. Leon Price received a DBA from the University of Oklahoma, returned as a faculty member, and has received several research and service awards as well as more than 25 teaching awards during his tenure. He was co-PI for an NSF grant and has received other teaching and research grants. He was involved in developing the MIS program at OU, developed the MIS field project course in the early 1980s, and has taught the course for over 25 years. He started working in information systems while in the U.S. Navy in 1958 and has continued his lifelong love of information systems for over 45 years. His background with the Federal Government and industry gives him a unique insight to information systems project development and implementation. His work as a vice president of a consulting firm provided insight toward outsourcing and prototyping as well as preparing RFP responses.

Professor Price's articles have been published in the *Academy of Management Review*, *Journal of Systems Management*, *Journal of Microcomputer System Management*, *DATA Management*, *Information Resource Management*, *Information Executive*, *Behavioral Science*, *Journal of Purchasing*, *Materials Management*, and *The American Oil and Gas Reporter*. He is a member of the Journal of Microcomputer Systems Management Review Board, the Association of Information Technology Professionals, and the Association for Information Systems; was selected the industry contact for the International Conference on Information Systems, and served as MIS chairperson for the Decision Sciences Institute and as the faculty coordinator for the annual Oklahoma Business Conference.

Preface

Information Systems Solutions: A Project Approach addresses the issues involved in undertaking a project to develop a computer-based information system solution in response to a problem or requirements posed by a client. Every organization that exists in our modern society uses some type of information system to conduct its activities. Many organizations today depend on computer-based information systems. With the advent of relatively low-cost personal computers and the Internet, even the local bridge club and the family next door probably make some use of computer-based information systems. In short, computer-based information systems exist very broadly across our society.

While computer-based information systems automate or computerize much of our information handling, these systems themselves are acquired, operated, maintained, and used by people. Anyone who has acquired a personal computer and associated software realizes that the acquisition of a solution can involve many issues. Large projects present a more complex set of questions than smaller ones, but all projects raise similar issues. The process of acquiring and implementing computer-based information system solutions involves a number of often-complicated tasks that can require a large amount of knowledge and effort to complete.

The prevalence of problems associated with information system solutions over the years led to a broad variety of approaches, methods, and tools. Students at universities or educational locations tend to learn about information systems in a set of specialized courses. At a university, the information system curriculum might include one or more courses on programming, data modeling, process modeling, infrastructure, project management, and other areas. Other providers may offer more specialized courses, for example, network and database administration.

A disconnect exists between the set of separate courses that cover information system theory, principles, methods and tools, and the knowledge, skills, and experience needed to acquire an information system solution. To succeed at providing a “satisfactory” information system solution, a person must understand how to:

- *Integrate knowledge* from a wide range of sources that include the standard information systems topics and also the topics that relate to the content of the environment for the system, for example, strategy, marketing, finance, accounting, production, organizational behavior, new product development, engineering, and science.
- *Apply knowledge* to solve a problem. Knowing about 10 different data modeling tools can help, but knowing how to select a tool that works and apply it correctly provides much more help.

While an ideal approach for learning to integrate and apply the knowledge and skills for information system solutions may not exist, the authors believe that this book can guide students toward an effective approach.

LEARNING MODEL

Information Systems Solutions: A Project Approach focuses on the issues of integrating and applying the knowledge and skills required to provide an information system solution. The basic learning mechanism consists of a student or team of students using the text as a guide to conduct an information system solution project. “Learning by Doing” in a structured project context offers the most satisfactory way yet found to learn about information systems solutions. After teaching literally thousands of university students in a course built around this text, the authors have received extensive feedback from both employers and employees who are former students that indicates the approach works exceptionally well. A learning model in a graphical format for a project course appears in Figure P.1.

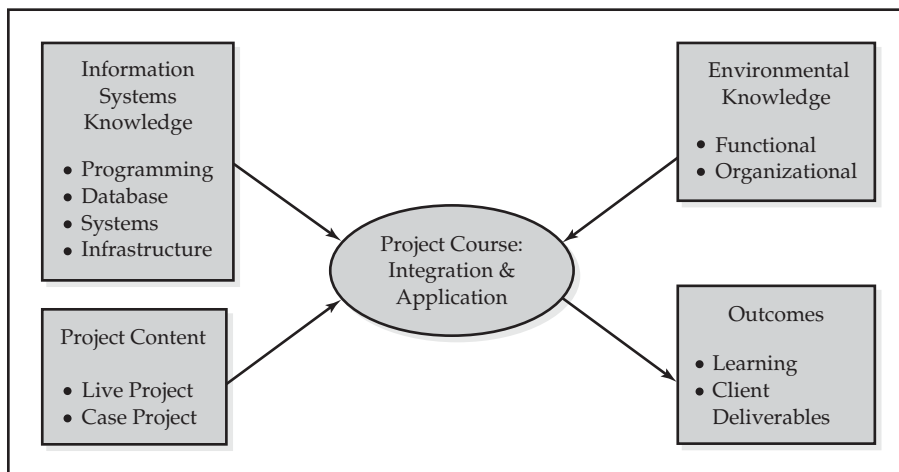
Information system knowledge may cover the following:

- **Programming.** The concepts and use of a programming language and preferably several, including Visual Basic or a similar language.
- **Database.** The concepts and use of entity relationship diagrams, relational schema, normalization, metadata, and a simple relational database engine, for example, MS Access.
- **Systems.** Systems analysis theory, process models, data flow diagrams, cost benefit analysis, prototypes, and project management.
- **Infrastructure.** Hardware and software basics, interoperability, and selecting hardware and software.

Environmental knowledge may cover topics that include:

- **Functional.** The basics of accounting, economics, human resource management, finance, marketing, production, quantitative methods, and strategy.

FIGURE P.1
A Learning
Model for a
Project Course



Some projects take place in specialized technical areas where the environmental knowledge may include areas of science and engineering.

- **Organizational.** The basics of managerial and organizational behavior.

The project-based learning experience may occur in a “capstone” course or in a two-semester “systems” course sequence if students have not taken system analysis and database courses. Students who possess all or most of the information system knowledge previously described, can either omit or review Chapters 4 and 5 of the text covering information system modeling tools. Students without a management background are advised to read an introductory management textbook. Most people working in organizations will possess the relevant environmental knowledge.

Project content represents a critical input to an effective project course. To a large extent, the project content determines the specific activities that the team undertakes. A good project contains enough data, process, and/or other content to challenge the student’s people, analysis, design, and project management skills. At the same time, a good project needs to be small enough to allow the students to complete the project within a semester or other appropriate time.

This book will work with three different project types:

1. **Case project.** Students receive a document or case that contains all of the information needed to carry out the course work in this text. The instructor’s manual contains cases that an instructor can use if desired. The instructor or an assistant can take the role of the client during student presentations and other interactions.
2. **Passive live project.** A project based on an actual organization but without any or much interaction with the people in the organization. For example, many Web-enabled applications, such as reservation and catalog sales, systems contain enough information on the Web to serve as passive live projects.
3. **Active live project.** The people working in an actual organization, for example, in a department or branch at a college, university, company, nonprofit, or government office, define the project and interact with the students.

Active live projects may offer some additional learning experiences in that students learn to work with a client and deal with ambiguity and change. With a limited supply of active live projects, an alternative is to assign the same project to all the project teams and let the teams compete for the best solution. Teams find learning from competitions stimulating and valuable especially when the judges include a senior person from the organization.

The project course may produce two outcomes. The primary outcome consists of student learning. A typical course requires the students to apply more structure with methods, tools, reports, and presentations than teams would use for any but the largest projects in an actual work environment. The extra structure enhances the transferability of the learning experience from small class projects to the wide variety of projects encountered in the work environment. For active live projects, the students also produce deliverables for and receive feedback from the clients, activities that provide important learning experiences. Many organizations use the systems or the recommendations the teams deliver.

ORGANIZATION AND CONTENT

The organization of this book follows the System Development Life Cycle (SDLC), a widely accepted project management structure. The materials in the text are arranged and grouped to appear where and when the students need them. All the sections contain numerous examples and guidelines for application of the text materials to a project. Project management concepts and tools are applied throughout the book, as they should apply throughout a project.

The first section, “Project and Team Organization,” prepares students to work effectively in teams, to understand system process and data concepts, and to manage a project. Even a student working by him- or herself during the course project can benefit from the material about working in teams because many projects in the work environment use teams. As noted earlier, students who have taken systems analysis and database courses can omit or briefly review Chapters 4 and 5. The material contained in Chapters 4 and 5 is applied and referenced often in the remaining chapters.

The second section, “Project Definition,” guides students through interacting with the client to learn about the organization’s strategy and the features and constraints for the proposed system. In many situations, the students also will want to learn about and document the key aspects of the current operation. Since misunderstandings between analysts and clients plague project work, the text explains a formal process for the students to review their understanding of the project with the client.

The third section, “Proposed System,” helps students to identify the data, processes, and physical and organizational infrastructure specifications for the proposed system and to document the specifications in both text and graphical formats. Students face a major decision at this point: select an alternative to develop during the System Delivery phase of the project. The alternatives may involve combinations of purchasing or building a solution and different levels of functionality.

The final section, “System Delivery,” provides guidance for the tasks required to either purchase or build a solution. Since the build and purchase tasks and issues differ, the topics are covered in separate chapters. A proof of concept model—an operational model of the solution—can strengthen greatly client understanding and confidence. The proof of concept model normally consists of an actual or demonstration version of a purchased solution or a prototype for a build solution. The last part of this section covers the tasks associated with compiling documentation including testing, training, implementation, and maintenance plans and with providing the final deliverables to the client.

Acknowledgments

Professor Price developed and implemented the University of Oklahoma project course in 1982. Professors Schwarzkopf and Van Horn have evolved and taught the course over a number of years. More than 3,000 undergraduate and graduate students have completed the course and over 200 organizations ranging from large corporations to very small nonprofits have served as clients. *Information Systems Solutions: A Project Approach* evolved from these teaching experiences and from the experience the authors gained working in and managing a large number of information system projects.

The authors express their thanks and gratitude to the many students and clients who have identified endless problems and contributed a multitude of good ideas. The corporate CIOs and other senior IT members of the OU Center for Management Information System Studies helped the authors to keep the text consistent with industry issues and best practices. Our colleagues at the University of Oklahoma, especially Professors Traci Carte and Jon Jasperson, provided insight on many issues over the years. OU graduate student Trevor MacCann converted the design specifications for the GB Video case into a proof of concept model in Microsoft Access. We thank Dennis Logue, Dean of the Price College of Business and Robert Zmud, Director of the MIS Division for supporting the project course.

Brief Contents

PART ONE

- Project and Team Organization 1**
- 1** Introduction to the Project Approach 3
 - 2** Organizing and Working in a Project Team 33
 - 3** Project Management 63
 - 4** Data Modeling 113
 - 5** Process and Object Modeling 155

PART TWO

- Project Definition 191**
- 6** Understanding the Client's Problem and Organization 193
 - 7** Learning from the Current Situation 229

PART THREE

- Proposed System 253**
- 8** Proposed System Specifications 255
 - 9** Alternatives, Evaluation, and Recommendation 297

PART FOUR

- System Delivery 337**
- 10** Outsourcing 339
 - 11** System Design 373
 - 12** Proof of Concept 423
 - 13** Project Completion 455

APPENDIX

- GB Video Final Report 481**

Contents

PART ONE PROJECT AND TEAM ORGANIZATION 1

Chapter 1

Introduction to the Project Approach 3

Introduction 4

The Historical Role of Information 4

A Typical IT Project 5

Information System Solutions 5

System Solution Activities 6

Concepts and Models for Information Systems 8

Roles for Information Systems 12

The Information System Life Cycle 12

Adding Structure to System Acquisition 14

Project Management 15

Systems Development Life Cycle 16

Balancing Structure and Flexibility 20

Performing Development Activities 21

Technology-Driven Development 21

Output-Driven Development 22

Data-Driven Development 23

Process-Driven Development 24

Event-Driven Development 25

Object-Oriented Design 26

CASE Tool-Driven Development 26

Structuring Development Activities

in an IT Project 27

Field Project Challenges 27

Summary 28

Key Terms 29

Review Questions 30

Critical Thinking Exercises 30

Individual Exercises 30

Group Exercises 31

References 31

Chapter 2

Organizing and Working in a Project Team 33

Introduction 33

Team Theory and Principles 35

Building an Effective Team 36

Start Up 36

Team Evolution 36

Team Contract 39

Skills Inventory 39

Assigning Roles 41

Code of Conduct 42

Managing a Team 43

Successful Teams 47

Individual Needs 48

*Leadership from a Motivational
Perspective 50*

Working in a Team 51

Communication 51

Manager Relations 52

Dealing with Nonperforming Members 53

Removing a Member 54

Resignation 55

Dysfunctional Teams 55

Peer Evaluations 57

Summary 59

Key Terms 61

Review Questions 61

Critical Thinking Exercises 61

Individual Exercises 61

Group Exercises 61

References 62

Chapter 3

Project Management 63

Introduction 64

Project Planning 64

Using the SDLC for Planning 65

The Spiral Model for Project Planning 66

Flexible Project Planning 67

Planning Mechanisms 69

Generating the Plan 73

Statement of Work 80

Project Execution and Control 84

Project Execution 85

Controlling Changes in Operations 85

Controlling Changes in Requirements 86

Monitoring Progress against the Plan 87

- Taking Corrective Action* 88
- Project Review Points* 89
- Project Management Tools 90
- Project Communication 91
 - Progress Reports* 91
 - Written Reports* 92
 - Presentations* 98
 - The Final Presentation* 103
- Summary 104
- Key Terms 107
- Review Questions 107
- Critical Thinking Exercises 107
 - Individual Exercises* 107
 - Group Exercises* 108
- References 111

Chapter 4

Data Modeling 113

- Introduction 113
- Entity Relationship Data Modeling 114
 - Model Components* 115
 - Entity Relationship Diagram Symbols* 117
 - Building a Simple ERD* 118
 - ERD Rules* 122
 - Additional Constructs for ERDs* 124
 - Simplified, Reduced-Form ERDs* 130
- Conceptual Data Models 131
- Metadata 133
- Enterprise Data Models 133
- Logical Data Models 136
 - The Relational Model* 136
 - Relational Schema* 140
 - Normalization* 144
 - Structured Query Language* 145
 - Dimensional Models* 146
- Summary 147
- Key Terms 150
- Review Questions 150
- Critical Thinking Exercises 151
 - Individual Exercises* 151
 - Group Exercises* 152
- References 153

Chapter 5

Process and Object Modeling 155

- Introduction 156
- Process Models 157
- Data Flow Diagrams 157
 - DFD Symbols* 158
 - Building a Simple DFD* 159
 - DFD Rules* 162
 - Creating Hierarchical DFDs* 165
- Other Process Models 174
 - IPO Charts* 174
 - Process Hierarchy Charts* 175
- Object Models 176
 - Use Case Diagrams* 178
 - Class Diagrams* 181
 - Sequence Diagrams* 183
 - Advantages of Object-Oriented Design* 184
- Summary 184
- Key Terms 187
- Review Questions 187
- Critical Thinking Exercises 188
 - Individual Exercises* 188
 - Group Exercises* 189
- References 190

PART TWO

PROJECT DEFINITION 191

Chapter 6

Understanding the Client's Problem and Organization 193

- Introduction 194
- Strategic Alignment 195
 - The Organization Case* 196
 - Determining Alignment* 198
- The Project Definition Report 202
 - Project Statement* 203
 - Strategic Alignment* 204
 - Proposed System Features* 205
 - Constraints* 207
 - Scope* 209
 - Examples of Project Definition Materials* 210

Working with the Client	210
<i>Professional Behavior</i>	215
<i>Prepare for a Visit</i>	216
<i>Make a Visit</i>	218
Information Collection Approaches	220
<i>Interviews</i>	221
<i>Group Interviews</i>	221
<i>Documents</i>	222
<i>Observation</i>	222
<i>Surveys and Sampling</i>	223
Summary	223
Key Terms	226
Review Questions	226
Critical Thinking Exercises	228
<i>Individual Exercises</i>	228
<i>Group Exercises</i>	228
Reference	228
Chapter 7	
Learning from the Current Situation	229
Introduction	229
Information Collection	231
Current Situation Narrative Model	232
<i>Description of Current Operations</i>	233
<i>Physical and Organizational Infrastructure</i>	234
<i>Problem Analysis</i>	234
<i>Retention and Change Analysis</i>	234
<i>Correctness and Completeness with Multiple Representations</i>	239
Current Operation Graphical Process Model	240
<i>Guidelines</i>	240
<i>Process Model Metadata</i>	243
Current Operation Graphical Data Model	243
The Project Definition Presentation	247
Completing the Project Definition Stage	249
Summary	250
Key Terms	251
Review Questions	251
Critical Thinking Exercises	252
<i>Individual Exercises</i>	252
<i>Group Exercises</i>	252

PART THREE

PROPOSED SYSTEM 253

Chapter 8

Proposed System Specifications 255

Introduction	256
Goals and Outcomes	257
Concepts for the Proposed System	258
<i>Problem-Solving Methods</i>	259
<i>Organizational Models</i>	263
<i>Design Approaches</i>	264
Narrative Specifications	266
<i>Narrative Format</i>	266
<i>GB Video Narrative Model</i>	269
Graphical Process Specifications	269
<i>Modified Data Flow Diagrams</i>	269
<i>The Context-Level DFD</i>	275
<i>The First Explosion MDFD</i>	275
<i>Additional Explosions</i>	278
Graphical Data Specifications	278
Metadata Specifications	281
Object-Oriented Design Specifications	282
<i>Use Case Diagrams</i>	287
<i>Class Diagrams</i>	288
Summary	291
Key Terms	293
Review Questions	293
Critical Thinking Exercises	294
<i>Individual Exercises</i>	294
<i>Group Exercises</i>	295
References	295

Chapter 9

Alternatives, Evaluation, and Recommendation 297

Introduction	297
Making Choices	299
Alternative Solutions	300
<i>Choosing a Design Option</i>	300
<i>Choosing Functionality</i>	301
<i>Choosing a Sourcing Option</i>	302
<i>Choosing Infrastructure</i>	307
<i>Evaluating Performance</i>	308
<i>Describing Alternative Solutions</i>	308

Evaluation 309
 Feasibility 311
 Risk Analysis 313
 Cost/Benefit Analysis 315
 Evaluation Metrics 319
 Features Analysis 324
An Example of Alternatives 324
The Evaluation Comparison and
the Recommendation 329
 The Recommendation 329
 Client Approval to Proceed 332
Summary 332
Key Terms 333
Review Questions 334
Critical Thinking Exercises 335
 Individual Exercises 335
 Group Exercises 335
References 336

PART FOUR

SYSTEM DELIVERY 337

Chapter 10

Outsourcing 339

Introduction 340
The Outsourcing Process 341
 Models of Outsourcing 341
 Determining Requirements 343
Product Features 344
 Functional Features 344
 Operational Features 344
Vendor Roles 351
 Vendor Features 352
 Product/Vendor Selection
 Issues 353
Request for Proposal 354
 RFP Content 355
 GB Video RFP Example 356
Features Evaluation 363
 Ranking Methods 363
 Identifying Candidate Solutions 365
 Assigning Ratings 365
 Outcomes 367

GB Video Example of a Weighted
 Features Analysis 367
 Contracts 367
Summary 367
Key Terms 369
Review Questions 369
Critical Thinking Exercises 370
 Individual Exercises 370
 Group Exercises 371

Chapter 11

System Design 373

Introduction 374
A System Design Framework 375
 Physical Infrastructure 377
 Organizational Infrastructure 378
 Infrastructure Example 379
Specifying Data Structure 379
 Relational Schema 381
 Metadata 382
 Other Data Schema 386
Specifying Processes 387
 Program Structure Charts 387
 Physical Data Flow Diagrams 389
 Process Model Metadata 389
Module Design 390
 Module Specification 391
 Pseudocode 392
 Metadata for Module Logic 396
 TIPOT Charts 399
Dialog-Driven Systems
Design 399
 Page Navigation Maps 402
 Page Action Maps 404
 Page Navigation and Action Map
 Metadata 407
Data Warehouse Design 408
 Dimensional Models 408
 Data Warehouse Metadata 411
 The Extraction-Transform-Load
 Process 411
Summary 418
Key Terms 419
Review Questions 419

Critical Thinking Exercises	420
<i>Individual Exercises</i>	420
<i>Group Exercises</i>	420
References	422

Chapter 12

Proof of Concept 423

Introduction	424
Types of POC Models	425
<i>Package POC Models</i>	426
<i>Prototype POC Models</i>	428
Using a POC Model	429
<i>Evaluating Operational Feasibility</i>	429
<i>Evaluating Design Parameters and Compatibility</i>	430
Prototype-Based Design	431
Building a Prototype	434
<i>Choosing a Focus</i>	434
<i>Making Initial Design Decisions</i>	436
<i>Generating Code</i>	437
<i>Schedules and Assignments</i>	437
<i>Coding and Design Specifications</i>	438
A GB Video Prototype	440
<i>Creating the Tables</i>	441
<i>Coding the GB Prototype</i>	445
Summary	452
Key Terms	453
Review Questions	453
Critical Thinking Exercises	454
<i>Individual Exercises</i>	454
<i>Group Exercises</i>	454

Chapter 13

Project Completion 455

Introduction	455
Testing Plans	456
<i>Desk Checks</i>	457
<i>Walk-Through Tests</i>	457
<i>Design Specifications Walk-Through Tests</i>	459
<i>Operational Testing</i>	459
<i>Post-Implementation Tests</i>	460
<i>Documentation Clearance</i>	460
Implementation	461
<i>The Implementation Plan</i>	461
<i>Implementation Strategies</i>	462
<i>Training</i>	465
<i>Maintenance Plan</i>	469
<i>Documentation</i>	469
<i>System Controls</i>	471
<i>Disaster Plans</i>	471
<i>Post-Implementation Audit Plan</i>	472
<i>GB Video Implementation Plan</i>	472
Closing the Project	474
Summary	474
Key Terms	478
Review Questions	478
Critical Thinking Exercises	479
<i>Individual Exercises</i>	479
<i>Group Exercises</i>	479
Reference	479

Appendix

GB Video Final Report 481

Index 511

Project and Team Organization

Over a lifetime, a person will participate in thousands of projects. A project consists of a problem to be solved in a defined time period. Projects can include such activities as buying a car, building a house, or choosing a college to attend. Sometimes people carry out projects by themselves; at other times, several people work together in a team on the project. As projects grow larger, the use of teams becomes more common.

This book addresses the issues involved in finding solutions to information system problems. A person in an organization identifies an information system problem and wants a solution. Often, the organization sets up a project to find a solution and assigns a person or a team of people to do the work. The project starts, the team works on finding a solution, and the project ends when the team recommends or implements a solution or when the available time or other resources run out. The five chapters in this section address some key areas of underlying skills and theory for finding information system solutions including organization theory for teams; project management; and data, process, and object modeling.

Chapter 1 explores the concepts and issues of using a project to solve an information system problem. Most learning takes place in a one-way, single-discipline, focused context. The learner strives to acquire specific facts about a specific topic, for example, data modeling, network design, or the Java programming language. While this kind of learning works well for many subjects, problem solving in a team and project context requires additional skills. These skills can include identifying and planning activities, recognizing the specific knowledge needed from a wide range of topics at each point in the project, actually applying the integrated knowledge to work toward a solution, monitoring the progress of the effort, changing direction as required, and working effectively with other people. Chapter 1 builds the framework for learning to work in a project context.

Chapter 2 examines in depth the issues and skills for working on a project as part of a team. A project team consists of two or more people jointly responsible for reaching the goal of a successful project. Teams can reduce the time needed to complete a project and can improve quality by bringing a wider range of

Chapter One

Introduction to the Project Approach

Chapter outline

Introduction

The Historical Role of Information
A Typical IT Project

Information System Solutions

System Solution Activities
Concepts and Models for Information Systems
The Component Model
The Content Model
The Technology Level of Models
Roles for Information Systems
The Information System Life Cycle
System Acquisition
System Use and Refinement
Adding Structure to System Acquisition

Project Management

Systems Development Life Cycle
Project and Team Organization
Project Definition
Proposed System
System Delivery
Balancing Structure and Flexibility

Performing Development Activities

Technology-Driven Development
Output-Driven Development
Data-Driven Development
Process-Driven Development
Event-Driven Development
Object-Oriented Design
CASE Tool-Driven Development
Structuring Development Activities in an IT Project

Field Project Challenges

Summary

Key Terms

Review Questions

Critical Thinking Exercises

Individual Exercises
Group Exercises

References

INTRODUCTION

Every organization in modern society, whether part of business, government, school, or a not-for-profit or social group, requires an information system to function. Information systems enable organizations to detect and record status and events, both inside and outside the organizations, and to respond as appropriate. The information system at the local breakfast club may operate mostly through informal verbal communication among the participants. A large airline reservation system may utilize a number of large computers, thousands of user workstations, and extensive electronic communications. Many information systems, for example, systems for universities, utilize both computer-based components (e.g., accounting, budgeting, payroll, personnel, etc.) and noncomputer components (e.g., courses, textbooks, procedure manuals, information desks, etc.).

The Historical Role of Information

Information systems for organizations build on a long history. Society, civilization, and life exist in the presence of information processing. Whereas one often hears the expression *information age* used to describe the last 50 years, in truth, the information age began long ago with the first living organisms. Living things use information processing to survive and thrive, for example, to find food and to avoid danger. The most versatile information processor the world has ever known is a human being.

People use a wide range of **information technologies (IT)**, that is, mechanisms for information processing. As is true in many disciplines, people mimicked the information processing mechanisms of nature by developing language: the use of symbols in an organized pattern intended to communicate meaning. Early languages used both pictures and sounds for the symbols. Using language, people could communicate with each other in much the same way that the cells in their bodies communicated. With information processing via language, organized groups of people, such as schools, families, tribes, cities, and nations, developed.

As the information processing between humans evolved, larger vocabularies and written forms for languages developed. Writing overcame two major barriers to information processing: time and distance. Written messages could be preserved over time and carried over great distances. After several thousand years of the use of writing, printing was invented in the 15th century. The major significance of printing came from reducing the cost of information processing: Large numbers of people could now afford to access the accumulated wisdom of society.

Several hundred years later, in the 19th and 20th centuries, people learned to transmit symbols at the speed of light using electrical pulses, first with the telegraph and then with telephone, radio, and television. In 1950, the emergence of the electronic digital computer started a new round of advances in information technologies that continues today at an ever-increasing pace. For example, the Internet moved from its beginnings to near universal use in a decade. Half a century after the invention of electronic computers, electronic information technologies impact almost every aspect of our lives.

Our civilization and our lives depend on a history of advances in information technologies, including languages, writing, and the use of electronic technologies for processing information and for communication. Forecasting the future remains both difficult and uncertain. However, the history, universality, and power of information processing and future advances to it most likely will continue to position information systems as a significant factor in all of our lives.

A Typical IT Project

An information technology (IT) **project** is a set of related activities with specific beginning and ending times undertaken to create an information system solution for an organization. The acquisition of a computer-based information system solution may require a substantial investment in facilities, equipment, and people. This text provides a project approach to acquiring a computer-based information system solution to solve a problem for an organization.

During a typical project, the team of people responsible for the information **system solution** might perform the following activities:

- Organize the team and the project.
- Meet with the client and users to gather information about the project.
- Define the project—its scope, goals, constraints, possible solutions, development plan, and so on.
- Review the project definition with the client.
- Obtain client agreement on the work the team will perform.
- Determine detailed requirements for a solution.
- Generate and evaluate alternative solutions.
- Work with the client to select a recommended solution.
- Design the solution, build a prototype of it, and/or design the specifications for the purchase of a solution.
- Prepare implementation, training, audit, and maintenance plans.
- Deliver the recommended solution to the client.

In practice, project activities will vary greatly depending on the nature of the project, the client, the team, and the environment. For some student projects, the instructor or a teaching assistant may play the role of the client.

INFORMATION SYSTEM SOLUTIONS

Computer-based information systems in organizations do not just appear; people either formally or informally design and procure or build them using analysis and design skills and information technologies. Only a few years ago, most organizations built most of their solutions in-house using a staff of analysts, programmers, and other technology specialists. Today, however, most organizations purchase some and often many of their information system solutions from vendors.

The processes of analysis and design that lead to a new or modified information system solution are known as systems development or **systems analysis and design**. An information system solution replaces a **current situation** with a new or modified one. The current situation is the set of conditions that currently exist in the area of the organization under study. The current situation may or may not involve a current computer-based or manual information system. For example, a project may focus on a new system to perform functions that do not exist in the current situation.

System Solution Activities

The system solution process changes the information processing in an organization. An **organization** is any collection of people, processes, and other resources working together to accomplish a purpose or a set of functions. Organizations include business firms, nonprofits, government, and social groups. The changes in the information processing of the organization may affect computer programs, equipment and facilities, data stores, data input and output displays, the way that people do their jobs, products, markets, and the viability of the organization. Change or anticipated change always brings the potential for unexpected problems and disruptions.

A **client** is a person or group of people who represent the organization that will use the solution or new system. A client with control over funding and accepting a solution often is known as the **sponsor** for the project or system. The client may or may not be a **user** of the system and may or may not possess in-depth knowledge and skills for system development. Often a client will work with one or more IT staff people or system analysts who possess specialized skills for finding information system solutions. While on occasion one person working by him- or herself may develop a system, much systems development work utilizes **teams**. Teams can include IT staff, sponsors, clients, users, third-party consultants, and other specialists. The teams may identify projects, develop specifications for the proposed system, evaluate alternatives, acquire a solution, implement the system, and perform a myriad other activities as part of the solution process.

A team may follow a number of approaches to acquire a solution. From 1960 to 1990, teams generally built a solution, that is, developed the specifications and wrote the programs. At present, many teams and clients choose to purchase a solution. That purchase may involve (1) an existing package, (2) an information processing service, (3) a design and build service, or (4) some combination. The various ways to acquire a solution are known as **solution classes**. Each solution class will encompass a number of alternative solutions.

In many organizations, a subunit of the organization holds the responsibility for all or part of the acquisition, operation, and modification of information systems. This subunit may operate under the title of the **Information Technology Group (IT)**, the **Information Systems Group (IS)**, or a variety of other titles. The team members working on a project report to a manager, the person who has the responsibility for the team. The manager may work in the IT group or a functional group, such as finance, marketing, production, or sales. In a matrix organization, the team may report to several managers. An organization may contract for some

or all of the system solution work with a third-party **vendor**, a group outside the organization with specialized skills or products.

The system solution activities require knowledge and skills in many areas. What is needed is functional knowledge of such areas as finance, marketing, or production; technical knowledge of hardware, software, and related technologies; such design and development skills as analysis, synthesis, and problem solving; and such interpersonal skills as communication, motivation, and team building. People who wish to carry out system development activities face the challenge of ongoing reeducation. By most estimates, the knowledge and skills of a system analyst has a half-life of three years or less. In other words, half of their existing knowledge and skills lose relevance after three years.

The system solution process results in **deliverables**, which are products or outputs that are delivered to the client. The most important deliverable is the computer-based system that meets the client's objectives. Most projects produce other deliverables. The client and the analyst may work together to define the required deliverables. Deliverables in a project might include such elements as the following:

1. A contract and/or plan for the project—tasks, resource assignments, time frame, budget, and more.
2. A project definition—the strategic framework for the project; functions, features, and constraints for the proposed system; and an analysis of the current operations and problems.
3. Reviews of progress with the client.
4. Conceptual data and process requirements for a proposed system.
5. Logical and physical specifications for the proposed solution.
6. Evaluations of alternative solutions and a recommended solution.
7. An operational proof of concept model.
8. Computer programs, a package program, or a procurement contract for the recommended system.
9. Infrastructure specifications for the recommended system.
10. Testing plans.
11. Implementation and training plans.
12. Final system acceptance plan.
13. Post implementation review or audit plan.
14. Maintenance plan.

The system solution process often has a broad and major impact on the organization. The project, independent of outcome, will incur costs. A successful effort may bring benefits, including reduced costs, increased sales, and revenues; an improved competitive position of the organization, and so on. An unsuccessful effort can incur large costs with little or no benefit and may reduce the competitive advantage of the organization. Because a project to acquire a new system can involve high complexity, high risk, and high benefits, the management of the project activities in the solution process attract and deserve thoughtful analysis

and execution. The discipline of **project management** consists of a body of knowledge and a set of tools for managing projects.

IT projects compete for resources with other IT projects and with many such different kinds of projects as marketing campaigns, new plant and equipment, and others. The senior managers of an organization want to fund projects that contribute to the strategic goals of the organization, perhaps profits, sales growth, or customer service. Organizations often have ranking processes that use return on investment, net present value, and similar measures to identify acceptable projects. Once a project is accepted and started, the organization may hold periodic reviews to decide actions for the next period. These actions can include expanding, continuing, reducing, or canceling the project. In short, projects may compete for resources over their entire existence. Resource and organizational constraints often require the team to follow a less than ideal development approach. The team may have to reduce the project scope or leave out functions to meet cost and time constraints. Team participants also may not pursue some promising alternatives because the client is not interested in them.

In many cases, the client and senior managers assign a budget and other resources to a project with little or no involvement by the team members. In other cases, a project team is asked to prepare a proposal for the project. In all cases, the team members need to think about **strategic alignment**—the relationship of the project to the values of the organization. The team members make many decisions every day, some of which may significantly impact the value of the project. A team that understands the strategic alignment of their project can make better decisions, can help the project to retain management support, and can increase the likelihood of a successful outcome.

Concepts and Models for Information Systems

The fundamental aspects of an information system are data, processing, and actions. **Data** consist of symbols arranged in a structure, format, or pattern. People use characters, numbers, and punctuation to communicate written data and use sounds for spoken data. Computers use electrical pulses to transmit data. Although the sets of pulses in a computer mean little to most people, the computer follows a fixed structure to recognize sets of pulses as characters and numbers. When the sender and receiver of data can agree that the data represent a particular thing or construct, then the data communicate *information* or meaning. For example, 13578, 2 is data. When placed in context, for example, “Video number 13578 has status = 2. Overdue,” the data have meaning and provide information. The goal of an information system is to transform or **process** data into information that will allow the system and/or the user of the system to take appropriate *actions*. The system or the user can use the information about the status of video 13578 to take an action, such as send an overdue notice to the person who rented the video. In summary, an information system collects, stores, retrieves, and transforms data into information, data that are useful for making decisions or taking actions.

A number of different models or **representations** can define or characterize information systems. A representation consists of symbols, conventions, and rules to convey meaning about and allow manipulation of a situation. Macro represen-

tations of information systems provide a broad high-level view. The component and content models offer two useful macro representations of information systems.

The Component Model

The traditional component or flow model views an information system as a set of components or subsystems that work together to perform such functions as collect or record the input data, process the data, store and retrieve data, and produce output data. The component representation follows the physical structure of an information system. In an actual system, the physical infrastructure consists of hardware and software components that execute the input, processing, storage, and output functions.

FIGURE 1.1
A Component
Model of an
Information
System

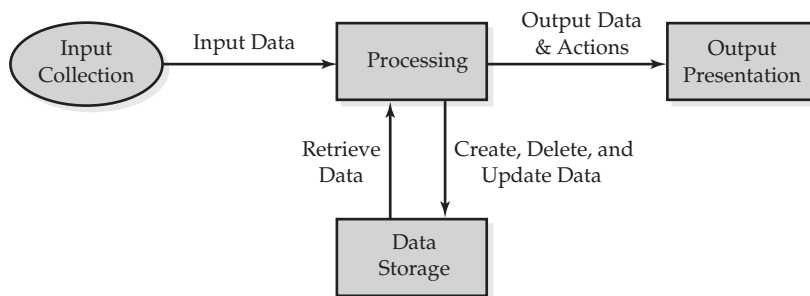
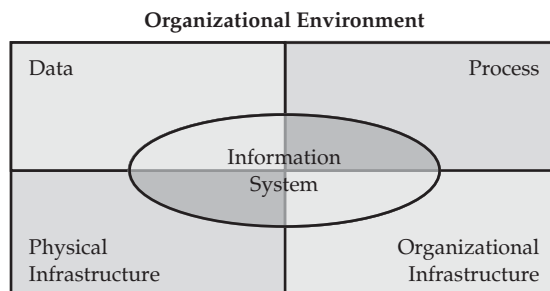


Figure 1.1 shows a graphical representation of the **component model** for an information system. In the component or flow model, data flow from one component to another is shown by the arrows. A number of widely used tools—data flow diagrams, for example—follow the structure of the component model.

The Content Model

For systems analysis and design purposes, the content representation of an information system shown in Figure 1.2 can provide useful structure and guidance. In the **content model**, an information system contains data, process, physical infrastructure, and organizational infrastructure. Each of the content areas interacts with all the others through the interconnections created by the system. As shown in the diagram, a specific system may contain within its boundary only part of the data, process, and physical and organizational infrastructure that

FIGURE 1.2
A Content
Model of an
Information
System



exists in the overall environment; the rest of the parts are external to the system, that is, outside the boundary.

The data section of the model describes the data structure for the system—the things or entities about which the system collects and uses data, the attributes of the entities, and the relationships between the entities. Most systems use only part of the enterprise data structure as shown in Figure 1.2, and several systems may use the same data. Entity relationship diagrams (ERDs) and relational schemas provide graphical tools to analyze and specify data structure in the content representation of an information system. A number of texts cover data modeling; for example, see Post, 2005 or Hoffer, Prescott, and McFadden, 2005.

The process section describes the processes that collect the input data, transform the data, store and retrieve the data, and generate outputs. Data flow diagrams (DFDs), function hierarchy diagrams (FHDs), and other process models or the computer programs themselves can represent the process content for a system. In good design, process and data models work together, and some representations of systems combine data and process. In DFDs, the analyst can use an ERD data model to define the data content of the data stores and then show processes and data flows in the process model. In object representation, objects describe both the data and process content of a system. Process models are covered in systems analysis and design texts, for example, see Whitten, Bentley, and Dittman, 2005 or Hoffer, George, and Valacich, 2005. For a discussion of object models, see Fowler, 2004 and Schneider and Winters, 2001.

The infrastructure of an information system defines all of the content other than data and process that is part of the system: computing facilities, buildings, people, organizations, and so on. For a discussion of infrastructure, see Burd, 2001. The **physical infrastructure** includes such things as the buildings or facilities, computing hardware (servers, workstations, networks, etc.), system software, operating systems, programming languages, and database engines that are used by the system. Parts of the infrastructure may support a number of systems. No standard graphical representations similar to ERDs or DFDs exist to model architecture. Projects can use variations of the component model to provide a pictorial or graphical representation of the infrastructure components and the network links. For an IT project, the team identifies the current infrastructure early in the project and later may recommend changes in the infrastructure to support the proposed system.

The **organizational infrastructure** defines the people, policies, and procedures that interact with the system. The overall strategy and policies for an organization form part of the organizational infrastructure for all the systems within the organization. For a video rental system, the organizational infrastructure includes the clerks who process the video rentals, the manager of the clerks, and the policies and procedures that apply to rentals. Issues include performance, authority, and responsibility for the system, for example, who specifies the requirements and/or pays for the system, how the system affects the functions and outcomes for the enterprise, who are the users, to whom do they report, who has access to and/or can change data, and who can approve or change the system. During implementation of the new system, *training* may represent a key and expensive

organizational issue. Standard representations exist for some of these relationships, for example, use cases, organization charts, and access control lists. Design and development tools, data dictionaries, and other system tools include provisions for including some organizational content.

In most education and training programs, separate courses cover the theory and applications in each of the content areas. The curriculum may contain one or more database courses, systems analysis courses, programming courses, infrastructure courses and such organizationally focused courses as strategy, organizational behavior, ethics, business law, finance, accounting, marketing, production and operations management. An IT project exercise requires the project team to *integrate the theory and applications* from each of the relevant content areas to create the new information system solution.

The Technology Level of Models

Information system representations also exist at different technology-specific levels. Standard levels include conceptual, logical, and physical. Different textbooks provide differing definitions for the meaning of these models. This book uses the following meanings.

- **Conceptual models** focus on the underlying content of an information system, for example, the data and processes in the system, independent of any technology or class of technologies. A use case, entity relationship diagram, or data flow diagram can represent a conceptual model since these diagrams need not imply the use of a particular technology. With conceptual models, technology does not place any constraints on the design.
- **Logical models** exist in the framework of a general class of technology, for example, data in the form of a relational database or processes in a procedural framework. Technology places constraints on “correct” logical models, for example, a relational model may not contain multivalued attributes or many-to-many relationships. A relational schema and a page navigation map represent logical models because they imply a class of technology, that is, a relational database and a Web-based application.
- **Physical models** include specific technologies, such as an Oracle database engine on a Hewlett-Packard server running UNIX or a Web site using Java. Physical models may include specifications for database engines, computer processors, input, output and storage devices, networks, operating systems, application programming languages, file partitioning, and more. Technology places many constraints on “correct” physical models, for example, interoperability, data formats, file sizes, and so on.

The dividing line between conceptual, logical, and physical models is not always clear or clean. For example, consider an ERD developed for an application on a relational database. While the ERD begins as a conceptual model, it becomes a logical model when the author includes specific features of the relational model, such as foreign keys. A process model for a system that includes operations specific to one vendor’s database engine becomes a physical model.

Roles for Information Systems

Organizations explicitly or implicitly apply the various information system models to create a wide variety of information systems. A one-person business may use an informal information system developed by the proprietor over years of experience and operated with the person's memory and a notebook or two. Large companies may use computer-based information systems with written policies and procedures, input forms, output reports, and complex computer processing programs. Just as people use their information processing abilities to see, hear, read, walk, talk, and carry out many other daily activities, organizations use information systems to direct and coordinate their actions, including ordering goods, serving customers, scheduling and controlling operations, arranging shipping, sending bills, and performing many other activities.

Information systems encountered in IT projects will play differing roles in the organization. Many IT projects involve administrative systems that support functional areas, for example, finance, accounting, payroll, and personnel systems. Others may focus on decision support systems that provide information to answer "what if" questions for analysis and forecasting. Some projects involve operational systems that handle physical activities, for example, order processing, making reservations, taking inventory, maintaining production control and shipping.

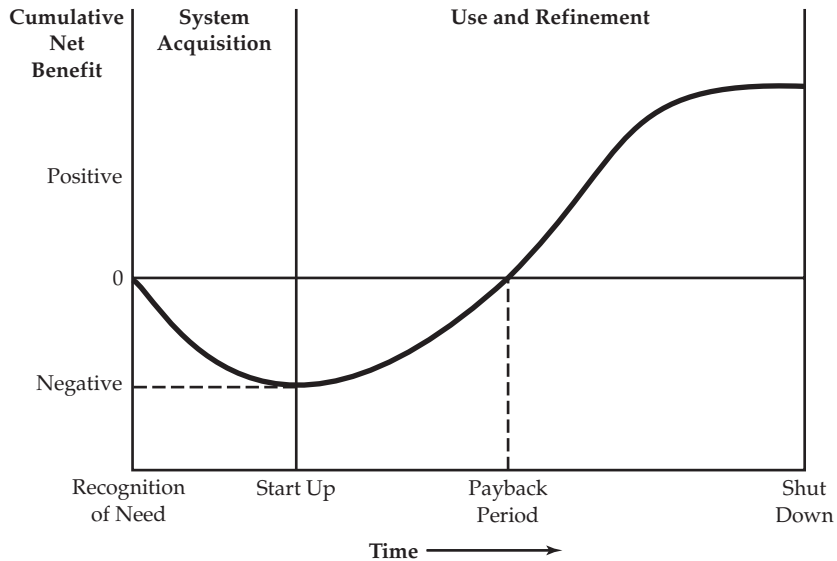
Many IT projects relate to selling products or services to customers. For example, when a retail business customer makes a purchase, the sales clerk may scan the identifier on each item of merchandise electronically to enter into the computer system the identification numbers of the items being purchased. The computer system uses the number to retrieve the name and current price of each item and then computes a total, adds taxes, and prints a sales receipt for the customer. At the same time, the computer system updates inventory, reducing the quantity on hand by one for each of the items sold and transferring the amount of sale and tax data to the accounting system. An IT project might create the sales system or create a decision support system to use data generated by the sales system.

IT projects can utilize a range of information technologies. Many IT projects focus on creating and using a database or a data warehouse. Recently many projects have involved the design and use of an Internet Web site. The Web site projects may link to a database. Some IT projects deal with processing transactions, including sales, purchases, payments, and more. Transaction processing system projects can present difficult challenges for completion within a semester and require careful scope bounds and planning.

The Information System Life Cycle

In common with most other mechanical and living systems, information systems experience a life cycle. They emerge, flourish, age, and eventually cease to exist. Information systems in organizations represent economic investments because the decision to create an information system is a decision to spend money with the expectation of future benefits. As illustrated by Figure 1.3, the information system life cycle consists of two major phases: (1) system acquisition and (2) system use and refinement. Each stage is shown in the figure and is described below.

FIGURE 1.3
The Information System
Life Cycle



System Acquisition

Acquisition starts with recognition of a perceived need for information and/or processing capabilities within the organization that are either new or a modification of those already in use. The desire for a new or modified system may result from a recognition of mistakes or problems with the current system or from changes in perceptions, people, organizations, products, markets, customers, competition, laws, or technologies that generate the desire for the new features. Whatever the reason, the recognition of the perceived need for a change marks the beginning of the information system life cycle. Acquiring a new system includes the steps of determining requirements and designing, building or purchasing, testing, and implementing the new information system. The organization hopes that the solution process will reach the start-up point: the time at which the new system begins to operate and possibly to generate benefits.

As noted, the organization initiates the solution process in anticipation of improving performance with respect to its strategic mission and goals. The acquisition of a system may take from several hours to a number of years and incur costs for people, hardware, software, and facilities. In general, no benefits accrue until the system begins to operate. *Marginal net benefits*, which are the marginal benefits minus marginal costs for each small time period of operation, may remain negative for some time after start-up until marginal benefits exceed the marginal costs. As a result, total net cost, or the negative value of the total net benefits for the system, will reach a maximum value at or sometime after the end of the acquisition period as shown in Figure 1.3. The time period of highest total net costs also represents a period of high risk in that a decision not to proceed with the system may result in a large loss for the organization.

System Use and Refinement

During the use and refinement phase, the organization uses the new information system. While the system operates, it may generate benefits, value, or revenues. Most systems also undergo significant refinement or modification after the period of use begins. The costs of refinement and modification generally exceed the original solution cost by a factor of two or more. Systems are created with the intent or hope that the total value or benefits generated by the system will exceed the total costs of solution, operation, and modification. Some clients focus on the payback period, which is the time at which the total benefits equal the total costs. Other clients may wish to find the system alternative that will yield the largest possible net present value (NPV), the time value of money adjusted sum of benefits minus costs over the life of the system or the best return on investment (ROI).

Although designers often plan for a new system to operate for three to five years, actual systems may remain in use for anywhere from a day to 20 or more years. This disparity between the planned and actual lifetime for systems led to the infamous Y2K problem. System designers assumed that their solutions would be shut down and replaced long before the year 2000 arrived so they allocated only two digits for the Year field. Unfortunately, many of these systems still operated as the year 2000 approached and had to be modified or replaced at great expense.

Eventually, the net marginal value of a system (1) will decline to the point where it produces little or no net benefit because the system outputs no longer are needed or effective or because further operation is too expensive; and/or (2) a possible replacement system is expected to produce larger net benefits. Changes in people, technologies, markets, products, law and regulation, competition, or the environment can cause the net benefit produced by the system to decline toward zero or below that of possible alternatives. If the system still produces a small net benefit, the organization may continue to operate it for some time until a replacement system is ready to operate. However, even without a replacement, if the system starts to incur a negative net benefit (that is, if the marginal costs of continuing to operate the system exceed the marginal benefits), the system should be shut down immediately. In many organizations, systems with no positive net benefit continue to operate because no one takes the time to analyze the situation and shut them down.

Adding Structure to System Acquisition

Because system acquisition is complex and demanding, people involved with it have devised and experimented with a number of approaches. As of yet, no one has found the ultimate approach. Most current IT projects use a variety of approaches, methods, and tools. The nature and composition of the mix has evolved over time in response to learning, new organizational demands, and new technologies. Often the components of the mix are learned as separate subjects, for example, database analysis, process analysis, infrastructure design, business communications, and financial analysis. In an IT project, the team must integrate the separate components to produce the set of deliverables. For all of the above

reasons, deciding on a reasonable set of approaches, methods, and tools for a specific project presents a major challenge.

When people began to build computer-based systems in the mid-1950s, they applied the **just-do-it model**, customized and largely informal approaches invented by themselves or picked up from a colleague. The people involved built systems with whatever plans or steps they thought were appropriate. They might think about a problem, discuss it with colleagues, write part of a program, go out and ask questions, and then write some more—in short, do whatever seemed right at the time. Designers built many successful systems with this approach, and some continue to build many systems this way today. The just-do-it approach offers a high level of flexibility. Talented, experienced system people can and do tailor their activities to fit the situation at hand with a high level of effectiveness.

As organizations attempted to build larger, more complex systems in the mainframe computer environments, the just-do-it approach sometimes led to problems. These disappointing results caused teams to search for more organized approaches to system development that might reduce costs and problems. All of the new approaches shared the common feature of adding **structure** which included step-by-step procedures and rules sometimes embedded in graphical representations, to the system development process. The intended benefits of adding structure to the system solution process are to increase benefits and reduce risk by (1) benefiting from the accumulated wisdom and experience of previous design efforts; (2) gaining consistency; (3) reducing the skill, expertise, and luck required for a favorable outcome; (4) improving the effectiveness of the new system; and (5) reducing the future costs of system maintenance. However, these structured approaches can reduce the flexibility that works so well for highly skilled, experienced people in the just-do-it approach.

Analysts can apply structure to two broad areas in system solution. The first encompasses project management or the planning and management of the solution process—adding structure to select and identify project tasks, to combine them with resources in a time frame, to monitor progress, and to take corrective action. The second focuses on the performance of development activities, adding structure to the methods and processes of systems analysis and design.

PROJECT MANAGEMENT

The act of acquiring a new system requires a broad set of planning and management skills and activities. The Project Management Institute defines project management as “the application of knowledge, skills, tools and techniques to project activities to meet project requirements” (*PMBOK Guide*, 2004). Project management focuses on meeting client expectations within the available time and cost constraints. Project management starts with the initiation of a project; covers such topics as planning, executing, and monitoring of project activities; provides for taking corrective action as needed; and continues until the end or closing of the project.

Project management is both a very old and a very new discipline. People have managed projects for thousands of years to construct roads, buildings, and fortifications and to conduct battles, wars, hunting expeditions, and other activities. Modern project management was influenced by such factors as the development of the Gantt chart and the Program Evaluation and Review Technique (PERT). Information system project management underwent rapid change in the 1970s with the arrival of structured development approaches and has continued to evolve at a rapid rate.

Although all projects involve some form of project management, the effectiveness of the management varies. With just-do-it approaches, the project activities that were undertaken, the order of the activities, and the quality of the deliverables varied widely from project to project. As noted, team members might begin by trying to write a program. The team might learn that it did not know all of the requirements or even the definition and boundaries of the problem. Several people might duplicate the same work while no one worked on other parts of the project. Further into the project, the team members might learn that the new programs did not include a number of features that the clients wanted and liked. Projects might fall behind schedule or the cost might exceed the budget. Sometimes the computing equipment selected before the design of the programs turned out to be unsuitable for the final design. New systems were installed without adequate testing. A system that works correctly might be very expensive to modify and maintain. These kinds of experiences convinced a number of people that information system project management could benefit from more structure.

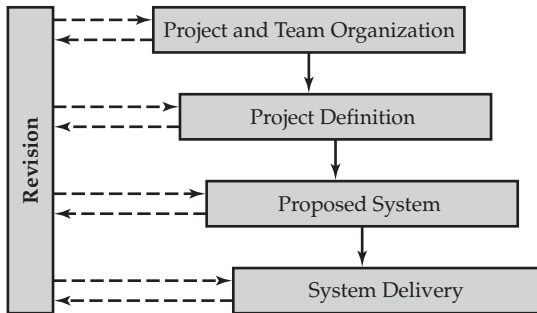
Systems Development Life Cycle

The **systems development life cycle (SDLC)** adds structure to the development process by (1) defining a set of uniform, explicit steps to be performed for all projects and (2) identifying a development time period that begins with identification of a problem that the new system might solve and continues through the design and implementation. In short, the SDLC concept matches the life of the development process to the economic life of the information system and thus explicitly recognizes that development continues throughout the life of a system. The SDLC concept provides both a description of the development phases and a prescription for a more effective development process. Various authors and managers use different steps and definitions for the SDLC, but all versions include defining steps that span the information system life cycle previously noted. In this text, the SDLC includes the steps described below that match the IT project plan used in the following chapters (see Figure 1.4).

Project and Team Organization

When a client requests a project, the team assigned to the project faces two immediate tasks: (1) organize the team and (2) organize the project work plan. No serious work occurs until the team makes a first pass at these steps. To organize, the team decides who will do what, how the team will operate, and what skills and resources the team can utilize. Chapter 2 discusses **team organization**.

FIGURE 1.4
The Systems
Development
Life Cycle
(SDLC)



The SDLC normally incorporates the waterfall concept: the deliverables from each stage flow into the next stage. Possible deliverables from this stage include a team contract and the initial plan and schedule for creating the new system. The plan potentially could include thousands of steps and can be revised at many points as the solution proceeds. Building a plan and managing the project is discussed in Chapter 3. Two basic tool areas for information system solutions, data modeling, and process modeling, are discussed in Chapters 4 and 5.

Project Definition

In the **project definition** stage, the team learns about the client's desires or requirements for the proposed system, expressed as features and constraints, and about the current situation. The client, manager, and team combine to make initial decisions on project scope. Project definition is discussed in Chapters 6 and 7.

The project definition stage normally starts with a strategic analysis of the organization, encompassing such questions as, What are the organization's and sponsor's values, mission, vision, goals, objectives, and performance measures? This strategic analysis, by telling the team what the organization values, thereby provides a framework for strategic alignment. That is, the team can align the project activities and solution design with the values of the organization. The team identifies the features that the client desires for the new system and also identifies constraints, for example, the budget to which the client will commit for the project.

The solution classes that the client will consider, for example, to fix the current system, to build a new system, to buy a package solution, and so forth, make up an important constraint. The team will create and evaluate specific detailed alternatives at later stages when the proposed system specifications are better defined. At this stage, the team just tries to understand the general type and nature of solutions that the client will consider. The team also checks the feasibility of alternatives and reduces or eliminates further work on alternatives that appear infeasible.

A second part of the project definition consists of understanding the current situation. The current situation analysis pursues a goal of identifying those aspects of the current solution that can contribute to the design of the proposed system. The analysis may produce a narrative description, enterprise data model,

and data flow diagrams for the current operations. The deliverables from this stage (strategic alignment, features and constraints for the proposed system, and text and graphical models of the current situation) flow to the next stage.

Proposed System

Once the client agrees that the team understands the project definition, the team begins to develop the specific conceptual specifications for a new system that will solve the client's problem. The team prepares a narrative model for the **proposed system** and the complementary graphical data and process models. For example, the team might prepare a conceptual data model, detailed data flow diagrams with explosions or other process models, and an extensive set of metadata. The goal is to make sure that the team fully understands and clearly documents the conceptual specifications for the proposed system. These activities are discussed in Chapter 8.

The team also expands the conceptual specifications into several alternative logical and physical solutions that appear to meet the client goals within the constraints, evaluates the alternatives, and attempts to select a recommended solution. These alternatives probably refine and extend some of the solution classes discussed earlier with the client. Evaluation forms a critical part of this stage. The analyst examines costs, benefits, advantages, disadvantages, and risks for each alternative, and then compares the alternatives and presents this information to the client. Risk analysis complements the economic analysis and examines the associated technical, operational, and organizational risks. The team may make a recommendation. Normally the client makes the final decision with input and advice from the team. Alternatives and evaluation are discussed in Chapter 9.

The deliverables from the proposed system stage—conceptual specifications, a description and evaluation of logical and physical alternatives, and a recommendation—go on to the next stage.

System Delivery

During the **system delivery** stage, the team, vendors, or other groups convert the conceptual design into a specific, detailed logical and physical realization of the recommended solution and prepare implementation and maintenance support plans. The exact form and nature of this stage depends on the recommended solution. If the team recommends purchasing a package or service, this stage may focus on preparing a request for proposal (RFP) to send to vendors and then to evaluate the responses. If the team decides to build the system, this stage will produce data schema and process logic and may produce a prototype and/or actual code. During system delivery, documentation becomes a critical issue. Any divergence between the documentation and the actual physical realization for the system can lead to serious problems during implementation and maintenance. Chapters 10 through 13 describe these steps.

The team can choose from a large number of tools and methods at this stage. The team may prepare a relational schema to define the table structure for the data, a process hierarchy chart (PHC) or a program structure chart to define graphically the structure of the program and a prototype to serve as a proof of

concept demonstration. As part of the associated metadata, the team provides pseudocode or actual code for each of the modules on the PHC. Other tools and methods used at this stage often depend on the choice of data management system and programming languages, that is, tools often come from the vendor or from a third party associated with the vendor and are specific to the language or data management system.

Testing forms a part of all aspects of the design phase and forms part of the basic plan for the development effort. Testing is used to answer two closely related but separate questions: (1) Does the computer-based system correctly perform the functions identified in the project specifications? and (2) Does the overall system meet the client's goals? Because development is complex and the client/analyst communication is at best inexact, an effective and comprehensive test plan can make the difference between a successful and a disastrous development effort.

The team may produce an implementation plan for use by the client or alternatively may use the plan to manage or carry out the implementation. The plan often will specify an implementation strategy and provide a detailed list of activities and the time schedule for each, often in PERT or Gantt chart form. These activities can include data conversion or collection, hardware and software acquisition, facilities construction, and training of customers, users, managers, and operators. Training represents a major task for implementation and one that often is underestimated.

The end of implementation marks the beginning of what often is the longest and most expensive period of the system life cycle, system operation, and maintenance. Problems in the design and changes in law, competitive strategy, management policy, and numerous other activities require maintenance that continues throughout the life of the system. Studies indicate that up to 80 percent or more of IT resources are used for maintenance activities as opposed to the initial solution. In view of this situation, inexpensive and rapid modification often become major goals of the system delivery stage. Building or purchasing a system that is easy to modify may offer far more benefits than trying to do everything right in the first version.

The final deliverables for this stage can include data schemas, process logic, documentation, a proof of concept model, programs, infrastructure specifications, and test, implementation, and maintenance plans.

As noted previously, the SDLC steps normally are arranged in a sequence or waterfall pattern as shown by the vertical arrows in Figure 1.4. The steps start with project and team organization and end with system delivery. The results and data from each stage feed into the one below. However, in practice, good systems development requires more flexibility. The horizontal arrows in Figure 1.4 indicate that the team may choose to skip stages or to return to an earlier stage, that is, go from any stage to any other stage either forward or backward to save time and effort, or to solve newly recognized problems or incorporate newly discovered ideas. The spiral model for project management in Chapter 3 offers another tool to reflect the need to recycle through the stages of the SDLC as the project proceeds.

Balancing Structure and Flexibility

Good information system development involves compromise and trade-offs. As noted, the SDLC adds structure and obtains a number of advantages from structure. Following all of the steps of the SDLC makes sense for the acquisition of a new, large utility billing system on a mainframe. For other situations, the SDLC may bring unneeded structure. For example, a client might request a minor process change from the analyst who maintains a system: to change the interest accumulation process on a class of bank accounts from monthly to daily. Current situation analysis offers no value—the analyst and client already know the current situation. The requirements are clear and need no analysis. The data structure is unchanged. The analyst can and should proceed directly to the design stage and insert the appropriate code. This approach certainly seems sensible, but is it consistent with the SDLC? Or alternatively, is the SDLC a rigid mandatory set of steps that all projects must follow or is it a checklist from which team members choose the parts they want? For large projects in the mainframe era, many IS groups viewed the SDLC as a rigid, mandatory set of steps and for these projects, the rigid view worked well.

Environment issues in the 1990s—rapidly changing business conditions, global competition, client/server and netcentric systems—focused new attention on issues of structure and flexibility. Relational database engines, package application software, and such fourth-generation languages as SQL, Visual Basic, C++, Java, and more, facilitated more flexible and evolutionary or trial-and-error type of development approaches. The emergence of client/server infrastructure facilitated decentralized development with smaller projects that focused on one area or a limited set of functions. The explosion of the Internet followed by intranet applications accelerated these trends. Increasingly, analysts and IT groups observed a mismatch between a rigid SDLC approach and the flexibility needed for rapid development of smaller projects. In today's environment, clients want the new application in days, weeks, or months; not after several years. With the new tools that are available, an analyst can create an application in days or weeks, but only by following a flexible set of steps. While good design can benefit from structure, it also clearly can benefit from flexibility—the freedom to tailor the approach to the problem.

Rapid development (RD) views the SDLC as a checklist from which the analyst selects the pieces that are needed for a particular project. The term *rapid development* is used here instead of the more common one of *rapid application development (RAD)* because RAD is sometimes associated with very specific development steps or tools. RD represents a concept, not a set of methods or tools. The highly flexible concept implied by RD is, Do whatever is necessary and/or appropriate to deliver an application (1) that meets the clients perceived needs; (2) in a short time; and (3) at a reasonable cost. With RD, the analyst, often in concert with the client, decides either explicitly in advance or implicitly by actions, which parts of the SDLC are relevant and how to achieve each part. Note that RD resembles the just-do-it approach except that it uses the SDLC as a starting point or checklist.

RD places a heavy responsibility on the analyst, client, and team. There is no cookbook to follow. It is up to the team to figure out how to perform the most important steps and to avoid everything else—a most challenging task. The team, often with client participation, reviews the problem or task and selects the steps that are needed to meet the RD goals of successful, fast, and cost-effective development. The client may provide guidelines, for example, informal plans for small projects and SDLC-like plans for very large, complex projects. In addition, the plan may be subject to review by a team manager.

The available resources may constrain the development plan. A team that believes that a new system offers the best solution may end up modifying the existing system because of time and resource constraints. Clients often prefer an acceptable solution now to a great solution at a much later date. With student projects, the semester time limit tends to restrict solutions. The team may not have enough time to learn new technologies and/or to include all the features that the client desires.

RD can reduce the development time, and experience suggests that keeping the development time as short as possible will reduce cost and will increase the probability of success. RD may work better than the traditional SDLC approach primarily because it can shorten the development time. Unfortunately, RD, along with most development approaches, methods, and tools, comes without any guarantees. Used incorrectly or inappropriately, RD can lead to problems, higher costs, and poor performance. As always, there is no substitute for thoughtful, knowledgeable analysts and clients.

PERFORMING DEVELOPMENT ACTIVITIES

Much of the time and effort in systems development goes to performing the activities defined in the SDLC or RD plan. Many of the problems that plague development also arise from performing these activities. In some circumstances, adding structure to the development activities themselves may alleviate some problems. From the earliest days of building information systems, designers have searched for effective ways to add structure to analysis and design, and a number of available approaches and tools do contain a significant amount of structure. Most projects can benefit from a combination of approaches and tools in response to such drivers of the development process as technology, data, and process. Accordingly, IT project teams experiment with a variety of approaches and tools.

Technology-Driven Development

Since data processing machines appeared, technology has influenced approaches to or provided structure for development efforts. Specific technologies tend to stimulate development related to the technology. For example, mainframes with magnetic tape data storage led to application-oriented development, while database engines have encouraged enterprise-oriented development. Relational databases, client/server, Internet technologies, and many programming languages have associated development tools and approaches. If an analyst plans to use a

relational database, the system often is created with the methods and tools recommended and provided by the vendor of the database. Oracle, for example, offers a set of tools that go with its database engine. These tools generate forms, reports, tables, and SQL code for the Oracle database engine; but they also promote an “Oracle data-centric approach” to the solution process. In any event, they provide a high level of technology-driven structure for the development effort.

Most modern system development tools structure development activities around technologies. For example much of the structure in the Oracle tools comes from relational algebra, SQL and the design of the Oracle database engine. IT directors tend to like technology-focused tools and find them effective. While **technology-driven development** can offer a number of benefits, it also may focus attention on technology features to the possible detriment of the client’s problems and goals. A technology focus on such tools as MS Access and Visual Basic tends to influence the design of the proof of concept model for the IT project. Virtually all projects involve at least some technology influence on design of the solution.

Output-Driven Development

Output-driven development, an approach that appeared in the 1960s, uses output to structure development activities. The analyst focuses the development activities on the outputs that the client and analyst desire from the system. The outputs in turn define the inputs, processes, and files required to obtain the outputs. Clearly, an output focus gives more attention to clients and organization issues than a technology focus. The structure consists of a set of rules or guidelines for (1) asking clients what they want as output from the system; and (2) deriving the inputs, files, and processing needed to create the outputs. Although output-driven development brought client and problem focus, the original versions lacked sufficient rules and structure to assure consistency and completeness. Additional structure was added by using matrices to depict graphically the relationships between outputs and inputs or outputs, processes, and inputs. In virtually all projects, the outputs desired by the client exert a major influence on the design of the solution.

Prototyping represents a modern form of output-driven development. With a **prototyping** approach, the team builds a working model of the actual system that produces the key outputs in response to actual inputs. The prototype may represent a simplified version of the system with only a small number of records in each file and with some complex processes and features omitted. The team can use the prototype in iterative fashion to determine and refine system requirements. The steps of the prototyping approach are as follows:

1. Create a first version.
2. Demonstrate the prototype to the client and/or user. Let the user actually use the prototype if possible.
3. Refine the prototype based on client/user comments.
4. Return to step 2.
5. When the client or users are satisfied with the prototype, build or procure the actual system.

Modern programming languages and development tools allow teams to build, and especially to modify, prototypes within reasonable time and effort constraints. While the original output models were descriptive, prototypes are operational, that is, they produce actual outputs for the analyst, client, and user to review and modify. A prototype also may reveal deficiencies in process, inputs, and files or tables. In common with most operational models, prototypes structure the development process more rigorously than the descriptive development approaches.

Data-Driven Development

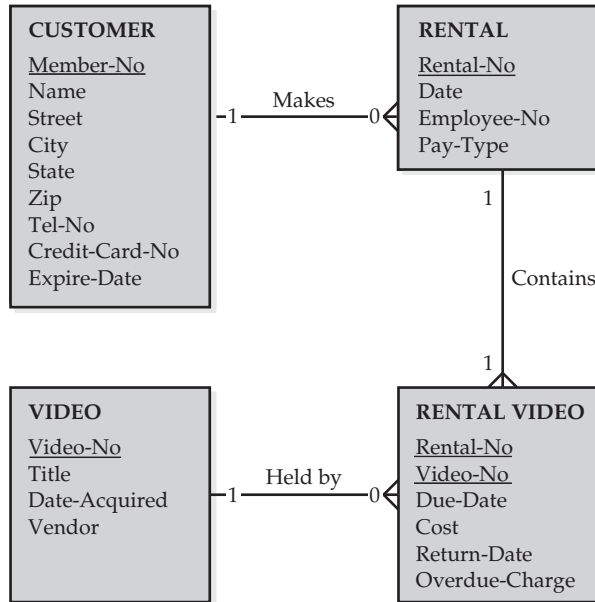
Data-driven development relies on a data model to structure development activities. The analyst focuses on defining the underlying structure of data. Some authors argue that data models are process and output independent and thus are the polar opposite of process and output-oriented development. The argument holds that once the underlying data structure is defined almost any desired output can be generated, and any process can use the data. In practice, process, data, and output do interact. The analyst has to think about desired outputs and processes to decide what data to include in the data structure.

In recent years, data and databases have formed the core of many information systems. If the analyst can identify, store, and access the appropriate data, processes to transform the data into desired outputs are relatively easy to create and modify. In addition, data seem to change less rapidly than the processes that use the data. A number of tools and techniques recognize the important role of data. Output-focused development and many process models explicitly deal with data. Entity relationship diagrams provide a well-structured graphical tool to implement data-driven development in a (somewhat) process- and output-independent form. The federal government has adopted standards called IDEF1 and IDEF1X (see <http://www.edef.com/idef1x.html>) for data modeling, and various vendors sell computer-assisted software engineering (CASE) tools to facilitate the use of these standard models for database design.

Entity relationship diagrams (ERDs) add structure to data-driven development by creating a model that (1) identifies the things about which the system will collect data; (2) identifies what specific characteristics are collected for each thing; (3) defines relationships between the data; and (4) displays this information in graphical form. The ERD representation of data encouraged the development of data structure independent of any specific application, that is, in an example of content representation in which data and process are separate. A number of different applications or sets of processes can use the resulting data structure. ERDs appear to add structure while offering a high level of flexibility and operationality and thus have gained widespread use. For the IT project, variations of ERDs are used both to help understand the current situation and to specify the new one.

An example of an ERD appears in Figure 1.5. In the figure the boxes represent the entities and the lines connecting the boxes represent relationships. The ERD is explained more fully in Chapter 4.

FIGURE 1.5
An Entity
Relationship
Diagram
(ERD) for GB
Video



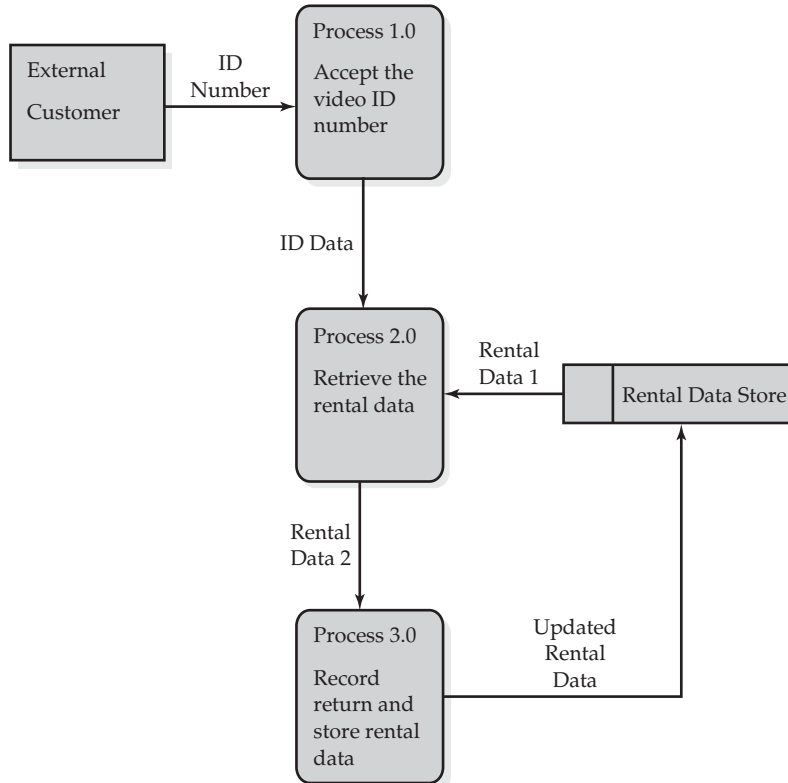
Process-Driven Development

Prior to the advent of effective data-driven development tools, many development efforts were process driven. The development activities focused on identifying and specifying the processes necessary to convert inputs to outputs. Process-driven development in common with data-driven development follows the content model of an information system but focuses on process rather than the data. Some process tools, for example, *process hierarchy charts (PHCs)*, graphically represent process largely independent of data. Using the PHC, an analyst can design the modular process structure for the system without explicit consideration of data.

Such process tools as **data flow diagrams (DFDs)** include both data and process and resemble the component model more closely than the content model. Data flow diagrams allow the systems analyst to study a system using the component representation of an information system where the components are input sources, output destinations, processes, data stores, and data flows. DFDs build on and add additional structure to the core ideas of output-driven analysis. Data flow diagrams give the systems analyst a consistent, well-defined, graphical tool to track data as it enters the system from an external source, moves through various processes, and is either stored or is sent to an external destination. The structure in the data flow diagrams comes from (1) decomposing a system into a set of basic components with defined behaviors; (2) following explicit and restrictive rules for data flows; and (3) using a graphical representation.

Figure 1.6 shows an example of a data flow diagram. The rectangles represent external data sources and destinations, the round corner boxes represent processes, and the arrows represent data flows. An additional explanation for this diagram appears in Chapter 5.

FIGURE 1.6
A Data Flow
Diagram for
GB Video



Analysts and organizations in the mainframe era used data flow diagrams extensively. Preparing DFDs can consume considerable time and effort, and the resulting DFDs may restate in a well-structured graphic form what the client and analyst already know. Some packages exist that allow the analyst to generate code from a DFD; without code, DFDs lack operability, which means the analyst cannot run a DFD to see how it works. With the emergence of fourth-generation languages (4GLs), relational databases, and network technologies, other process models and tools that facilitate code generation and bring more operability have gained increasing use. DFDs remain a promising communication tool in the IT project context for describing an existing system and for specifying a proposed system.

Event-Driven Development

Event-driven development reflects a change in the structure of the processes inside a computer-based information system. In a classic flow or batch system, a single trigger activates the system. For example, reaching the end of the billing cycle triggers a billing process. Once the process is triggered, control and data flow from one subprocess to another until the process ends. The subprocesses occur in a fixed sequence, namely, they (1) read the next set of billing input data; (2) retrieve the associated billing record; (3) process the data to compute the bill; (4) update the billing record; (5) prepare and print the bill; and (6) return to step (1). Unless interrupted by error conditions, the sequence repeats until the input data is exhausted.

With the advent of interactive systems, the flow pattern changes. In an interactive system, a series of external events trigger different subprocesses of the overall process. The subprocesses may have little direct relationship with each other and may occur in a wide variety of difference sequences and times depending on the external triggers. For example, a video store retail process might consist of major subprocesses for renting a DVD, returning a DVD, and enrolling a member. External events, such as a customer request to rent or to enroll, or a customer action to return a DVD, trigger the subprocesses associated with the events.

In event-driven development, the external events that trigger processes provide the structure for the development effort. The analyst focuses on identifying and understanding the events that trigger the subprocesses. Once the events are identified, the analyst can define the data and process logic associated with the occurrence of each event. DFDs work reasonably well to describe some event-driven systems, for example, video rental. Others, for example, Web site designs, are described better by such tools as page navigation maps discussed in Chapter 11. Once the system is described, the analyst may use event-driven and object-oriented program languages such as Visual Basic, Java, and SQL, to convert the design into an operational system.

Object-Oriented Design

Object-oriented design (OOD) focuses on and structures the development activities around objects. An object represents a thing, such as a person, product, or event, that has a well-defined role in the actual system. Objects contain or “encapsulate” both the data that describe the attributes of the object and the processes that describe the behaviors that the object can exhibit. A standard set of object-oriented diagrams and practices are defined in UML, the Uniform Modeling Language (see Fowler, 2004 for a description). In UML, some of the more commonly used diagramming and analysis tools include

- *Class diagram*—a diagram that resembles an ERD with a static representation of the data and operations (processes) associated with each object.
- *Use case*—a graphical representation of who does what in the system.
- *Sequence diagram*—a dynamic representation of the interactions between objects.
- *State diagram*—a dynamic representation of how the state of objects change in response to events.

In practice, the term *object-oriented design* is used by people to cover a range of design activities including much Web and most event-driven development, using such languages as Visual Basic, C++, and Java. In many of these efforts, the UML standard notation and diagrams are not used.

CASE Tool-Driven Development

Computer-assisted software engineering (CASE) tools for development consist of one or more computer programs to support the analyst’s activities. Everything from fourth-generation programming languages, including SQL and Visual Basic, to such comprehensive tools as Popkin System Architect and Oracle Developer Suite are described as CASE tools. Hundreds of CASE tools exist and are

in use. Such comprehensive CASE tools as System Architect (see http://www.popkin.com/products/product_overview.htm) and Oracle Developer Suite (see <http://www.oracle.com/tools/index.html>) contain mechanisms for creating conceptual and logical system models (e.g., ERDs, etc.) and also mechanisms to create or generate physical representations (e.g., databases, data tables, forms, screens, reports, queries and associated program code). Comprehensive or upper CASE tools incorporate aspects of process-, data-, output-, and technology-driven development and add a high level of structure to the development process.

CASE tools, and especially upper CASE tools, may increase the cost and time for system development. CASE tools work best when they are used frequently on projects. One-time or occasional use incurs high learning costs for limited benefits. The use of CASE tools also raises issues of continuing support for the CASE tool and the applications developed therein. When a vendor stops supporting a tool or an organization decides to discontinue its license for a CASE tool, maintaining applications written with the tool can become difficult and expensive.

Structuring Development Activities in an IT Project

As noted, all development activities use some structure. Because of their learning goals, student IT project development activities tend to contain more structure than comparable projects in most other organizations. In a typical IT project, ERDs and relational schema may specify the structure of data and DFDs, function hierarchy diagrams, and/or other process models specify the process and program structure. The FHD is complemented by further defining process with pseudocode, a structured language that follows clear rules. Not surprisingly, physical design contains the most structure of any phase of the solution process, and much of the structure is technology driven. Such tools as form, as well as report generators and programming languages add a high level of structure. Some general guidelines for Implementation exist, but implementation remains largely unstructured. The main structure in the implementation deliverable for an IT project may come from the use of text instructions and a Gantt chart.

FIELD PROJECT CHALLENGES

Field project teams face some formidable challenges: decide what, when, and how to do tasks and then do it all in a limited time. In an ideal world, a team applies a single tool or method to generate all of the required deliverables. In an IT project, the team, in common with system development teams around the world, must choose and integrate many skills, approaches, and tools. The portfolio includes SDLC, rapid development, interviews, reviews, narrative models, DFDs, ERDs, FHD, OOD, relational schema, pseudocode, 4GLs, and a host of other things.

Unfortunately, these approaches and tools may address the key issue only peripherally, that is, finding a solution that meets the client's perceived needs within the client's constraints. Finding a suitable solution generally will depend heavily on careful analysis and good creativity from the analysts accompanied by a lot of hard work. Of equal importance, information system projects involve organizational values and behavioral and political issues. The team must apply

communication, interpersonal, organizational, and political skills; strategic analysis; and the problem-solving and decision-making skills learned in many contexts. The team must then temper these skills with experience and common sense.

Every team faces a final challenge of doing a “good enough” job. One can always do more—more analysis, more alternatives, more evaluation, more design, more testing, and so on—but all at more time and expense. Everyone likes to talk about delivering the “best system” to the client. *Best* implies an optimal solution. In mathematics, the concept of optimization—finding a maximum or minimum of a function—has meaning. For information system solutions, even experienced and highly qualified teams cannot define the meaning of “an optimal system,” let alone find one. System development is characterized much better by muddling through the process than by optimization.

If the team members cannot expect to find the best or optimal solution, what should they strive for or expect to achieve? What is good enough? Often the team can avoid or eliminate a large set of clearly poor solutions: ones that are infeasible, too risky, or for which the costs far exceed the benefits. Teams can follow “best practices” to select approaches and tools that fit the problem and have a history of producing good results. Hopefully, the team can find one or more acceptable solutions that meet the client’s needs and constraints. In the end, the team should strive to find a good solution, the most promising of the known acceptable solutions. The purpose of this text is for the reader to learn how to increase the likelihood of finding and recommending a good solution or, at least, an acceptable one.

Summary

Every living thing and every organization require an information system to function. An information system collects, stores, retrieves, and transforms data into information, meaning data that are useful for making decisions or taking actions. The information system allows the organization to detect and record what is happening both inside and outside the organization and to respond as appropriate.

Many different useful representations exist for information systems. The Component model and the Content model offer two macro representations of information systems. In the traditional Component model, a set of components or subsystems work together to perform such functions as to collect input data, process it, store it, retrieve it, and produce output data. The component representation follows the physical structure of an information system; in an actual system hardware and software components exist that carry out each of these functions. For systems analysis and design, the Content model provides useful structure and guidance. In the Content model, an information system contains data, process, physical infrastructure, and organizational infrastructure. Each of the content areas interacts with all the others through the interconnections created by the system.

Information systems experience a life cycle. They emerge, flourish, and, as they grow older, possibly lose usefulness. The Information Systems Life Cycle consists of two major phases: (1) acquisition and (2) use and refinement. Acquisition starts with recognition of a perceived need for new or modified information and/or processing capabilities within the organization and ends at the point

the new system begins to operate. During use and refinement, the organization uses the new information system and modifies or maintains it. The organization hopes the new system will reach the payback point at which total benefits exceed total costs in a relatively short time.

The processes that lead to the solution for a new or modified information system are known as system development or systems analysis and design. System solutions replace a current situation with a new or modified one. A client is a person who has the authority and ability to order and/or pay for the new system. The system solution process results in deliverables—products or outputs that are delivered to the client. The system solution process often causes a broad and major impact on the organization. A successful effort may bring benefits and an unsuccessful effort may incur huge costs.

Because system solution is complex, adding structure to the planning, management, and performance of system solution activities can improve results. The System Development Life Cycle (SDLC) can add structure to the planning and management of the development process. In this book, the steps in SDLC comprise Project and Team Organization, Project Definition, Proposed System, and System Delivery.

A good system development plan allows a trade-off between adding structure and obtaining flexibility. While the traditional SDLC concept may limit flexibility, the rapid development version of the SDLC allows the team to do only those activities necessary to deliver an application that meets the client's perceived needs, in a short time, and at a reasonable cost. In the beginning, analysts built systems using whatever steps and approaches seemed appropriate to them. Over time, analysts used approaches based on technology, output, data, process, events, objects, and CASE tools to add structure to the performance of system development activities.

IT projects pose many challenges, for example, completing a number of inter-related activities in a limited time frame with limited resources. Team members must acquire and apply a number of technical and interpersonal skills. Finally, teams strive to arrive at a good solution, one that stands out from the known satisfactory solutions.

Key Terms

client, 6	deliverables, 7	object-oriented design (OOD), 26
component model, 9	entity relationship diagrams (ERDs), 23	organization, 6
computer-assisted software engineering (CASE), 26	event-driven development, 25	organizational infrastructure, 10
conceptual model, 11	information systems group (IS), 6	output-driven development, 22
content model, 9	information technologies (IT), 4	physical infrastructure, 10
current situation, 6	information technology group (IT), 6	physical model, 11
data, 8	just-do-it model, 15	process, 8
data-driven development, 23	logical model, 11	project, 5
data flow diagrams (DFDs), 24		project definition, 17
		project management, 8

Project Organization	strategic alignment, 8	team, 6
proposed system, 18	structure, 15	team organization, 16
prototyping, 22	system delivery, 18	technology-driven
rapid development (RD), 20	system solution, 5	development, 22
representation, 8	systems analysis and	user, 6
solution class, 6	design, 6	vendor, 7
Specifications	systems development life	
sponsor, 6	cycle (SDLC), 16	

Review Questions

1. What roles can a client and a systems analyst play in the system solution process?
2. What is a representation of a system?
3. Describe the information system life cycle.
4. Explain the differences between conceptual, logical, and physical models of a system.
5. Compare the component model and the content model for an information system.
6. What are the phases of the systems development life cycle, and how do they relate to the information system life cycle?
7. Define rapid development (RD). Why does or might it differ from rapid application development (RAD)?
8. Do team members normally create optimal systems? Explain your answer.
9. Who might approve system deliverables? Explain why you made your choice.

Critical Thinking Exercises

The critical thinking exercises ask the student to apply common sense with the tools and skills covered in this chapter. The Individual Exercises below are scaled to one person. The Group Exercises that follow provide suitable problems for a team of people working together. However, both sets of exercises will work for both individuals and groups.

Individual Exercises

1. Prepare a rapid development plan to acquire a university degree.
2. Describe the possible impact of a new computer-based system on a grocery store
3. The retail sale system for a store works as follows. The customer takes items to the checkout where the items are run by a scanner. The scanner or receiver reads the bar or radio code on each item and sends the code to a computer. The computer system uses the code to retrieve price and item description and at the end of the customer's order adds tax and computes a total. All this data prints out on the customer's receipt and is also stored in the system.
 - a. Draw a component model of the system.
 - b. Is your component model a conceptual, logical, or physical model?
4. For problem 3:
 - a. Draw a content model for the system.
 - b. Is your content model a conceptual, logical, or physical model?
5. Your team has been asked to transform a motor vehicle tracking system from an Access database to an SQL Server database. Which development driver will you use for the project? Justify your answer.

Group Exercises

1. Define the activities your team will coordinate with a client to prepare a rapid development plan.
2. A friend is getting married and has asked you for help in planning the big event.
 - a. Define the full SDLC steps required in planning a major wedding.
 - b. Your roommate decided to have a very small wedding. Using the RD approach, eliminate the unnecessary SDLC steps that you defined for the major wedding. Justify the steps you eliminate.
3. Answer the questions below for each of the following development drivers: Process, Event, CASE Tool, and Data.
 - a. Explain briefly the focus created by the driver.
 - b. What factors lead to or make the driver a good choice?
 - c. What are the consequences of a poor choice?
4. As an assistant soccer coach in the community little league, you learn that the records on team members, teams, schedules, referee assignments, and locations are in a three-ring binder maintained by the community activity director. You volunteer to build a computer system to track this data.
 - a. Which development driver will you use for the project? Justify your answer.
 - b. How or from where will you obtain the initial data for the computer-based system?

References

- Burd, Stephen D. *Systems Architecture*. Boston: Course Technology, 2001.
- Fowler, Martin. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. 3rd ed. Reading, MA: Addison-Wesley, 2004.
- Hoffer, Jeffrey A.; Joey F. George; and Joseph S. Valacich. *Modern Systems Analysis and Design*. 4th ed. Upper Saddle River, NJ: Prentice Hall, 2005.
- Hoffer, Jeffrey A.; M. B. Prescott; and F. R. McFadden. *Modern Database Management*. 7th ed. Upper Saddle River, NJ: Prentice Hall, 2005.
- PMBOK Guide: A Guide to the Project Management Body of Knowledge*. 3rd ed. Project Management Institute (PMI), 2004.
- Post, Gerald V. *Database Management Systems*. 3rd ed. New York: McGraw-Hill/Irwin, 2005.
- Schneider, G. and Winters, J. P. *Applying Use Cases: A Practical Guide*. 2nd ed. Reading, MA: Addison Wesley, 2001.
- Whitten, Jeffrey L.; Lonnie D. Bentley; and Kevin C. Dittman. *Systems Analysis and Design Methods*. New York: McGraw-Hill/Irwin, 2005.

Chapter Two

Organizing and Working in a Project Team

Chapter outline

Introduction

Team Theory and Principles

Building an Effective Team

Start Up

Team Evolution

Team Contract

Skills Inventory

Assigning Roles

Code of Conduct

Managing a Team

Successful Teams

Individual Needs

Leadership from a Motivational

Perspective

Working in a Team

Communication

Manager Relations

Dealing with Nonperforming Members

Removing a Member

Resignation

Dysfunctional Teams

Peer Evaluations

Summary

Key Terms

Review Questions

Critical Thinking Exercises

Individual Exercises

Group Exercises

References

INTRODUCTION

In a modern organization, analysts and designers undertake many projects as a member of a team. A project **team** consists of two or more people jointly responsible for accomplishing work on a project—the goal is a successful project. Teams may reduce the time needed to complete a project and may improve quality by bringing a wider range of experience and providing a potential way to use best the strengths of each team member. Most job recruiters rate the ability of a potential employee to operate effectively in a team as a critical information system (IS) skill. Working effectively as a member of a team requires both skill and constant thought. Intelligence, technical and business knowledge, and the willingness to work hard all contribute to becoming an effective team member, but success also requires an understanding of how teams function.

The sketches that follow illustrate some of the issues that people working as teams encounter. Each of the issues is developed later in this chapter.

Teams mature and change over time. Behavior that team members appreciate, or at least tolerate, at the beginning may provoke negative reactions as the team matures.

- In the first meeting of the project team, Alan took charge. Alan had very good grades and seemed to know what he wanted. For a while everyone felt like they were making excellent progress and told Alan so. By the end of the semester, however, Alan was complaining that he was doing all of the work. In the peer review at the end of the semester, Alan received a poor evaluation with most members calling him domineering and insensitive.

Teams form initial impressions of their members based on behavior during the first several meetings. These initial impressions can last throughout the project.

- Alice, a very bright student who does well in most of her classes, tends to be quiet and reserved. When the project started, Alice said little at the first meeting and did not request an initial assignment. She did offer to help wherever she was needed. The team assigned her to work on the team contract. Susan wrote the first draft and gave the draft to Alice to review. Alice, reluctant to appear critical of Susan, made no changes. When the team met to go over the final version, the members found a large number of errors and thought that the document dealt poorly with how to resolve disagreements. After this experience, the team did not ask Alice to do much and then gave her poor peer evaluations because of her lack of participation.

A good team member need not possess the credentials of a rocket scientist. Teams value members who contribute from the beginning to the end of the project and who work cooperatively.

- Jean is an average student, slightly introverted, and an excellent programmer. She attended all of the team meetings and volunteered to work on project definition and the prototype. She urged Ted to serve as the editor and checker for all their work because Ted writes well and is very careful about details. She volunteered to work with Ted. As she was building the prototype, Jean discovered a number of differences and problems with respect to the proposed system documentation. She talked to Ted and worked with him to bring the prototype and proposed system documentation into agreement. Jean received high peer evaluations from her team members.

Teams expect each member to make a reasonable contribution. Most teams will understand if a member faces constraints, such as a job or other classes. However, teams tend to look poorly on members who agree to do something and then do not do it.

- Edward, an excellent student, enrolled in 18 hours of classes and worked for the campus computer store. When the team met, Edward told everyone that he would appreciate plenty of advance notice for his assignments. At the next meeting, the team asked Edward to develop the data model for the proposed system

because he received an A in the Database course, and Edward agreed. Edward put off working on the task and missed a number of team meetings because of his other commitments. When asked about his progress, he reported he was on schedule. The night before the design was due, the store asked Edward to work overtime. He failed to complete his assignment, and his team lost points for a late report. Edward was surprised when he heard that the rest of the team had asked the team manager to remove him from the team for not contributing.

Some teams are effective and rewarding to work with; others are ineffective and dysfunctional. Much of the difference relates to how the team members manage their relationships with each other. A knowledgeable team member should recognize the issues in each of the previous sketches and know how to respond. This chapter presents some principles of team behavior that will help team members to recognize potential problems and deal with them. Effective team growth requires the knowledgeable cooperation of all of the members.

Most projects that can be completed by a class team in a semester are relatively small. Businesses might approach many of them as a single-person design task using prototyping with little planning or documentation. Because learning to practice effective team skills is one of the fundamental objectives of a field project course, the course assignments involve extensive analysis, design, and documentation even for small projects. Completing all of these activities requires coordinated work by the members of the project team. Organizing and managing a team is critical for success in the field project course.

TEAM THEORY AND PRINCIPLES

A team is a group of people, but not every group is a team. Greenberg and Baron (2000, p. 271) define a team as “a group whose members have complementary skills and are committed to a common purpose or set of performance goals for which they hold themselves mutually accountable.” Team characteristics for IT projects include the following:

1. A team contains two or more **members**, anywhere from tens to hundreds in a large project. A manager generally assigns members to the team. Team members may suggest the choice of other members but may not make the final decision. Prior to joining the team, the members may or may not know each other personally or professionally. To the extent that members possess limited information about each other, mechanisms for learning about each other’s abilities and for learning to trust each other are important. Teams in organizations often include members from both the IT and client communities. While including clients and users puts more of an organizational burden on the team initially, the combination of skills generally produces a better product for the client.
2. Team members share a common purpose or **goal**—to create a satisfactory IT product or system for the client within the constraints. Most team members will pursue personal goals as well, for example, to obtain a good evaluation by the manager or client, to work on other projects or tasks, to learn new skills, to find a better job, to lead a good life, and more. Sometimes personal goals

- conflict with the team goal due largely to conflicts over the use of time. One of the ways to improve the effectiveness of the team is to recognize legitimate individual goals and to support them within the context of the team objective.
3. Team members possess different skills. A well-functioning team makes assignments to take advantage of the best skills of each member. A team contains a portfolio of different personalities; in most cases, the composite ability of the team exceeds the ability of the best member.
 4. Team members are **mutually accountable** for the success of the project. If the project succeeds, the team succeeds; if the project fails, the team fails. The team may have an explicit or implicit structure to reward both total team and individual contributions to the project.
 5. Organization and management provide mechanisms to improve the **effectiveness** of a team. Even with the best of intentions, team members may find that working together effectively presents a difficult challenge.
 6. Team members need to **communicate** or share information with each other that is not readily available to nonmembers of the team. Mechanisms and technologies that facilitate communication may offer important tools to increase the effectiveness of the team.

Team organization and management consist of two stages: (1) working out initial roles and responsibilities for team members; and (2) managing relationships as the team matures to produce good results. Teams develop more quickly and evolve more positively if the team members organize and manage their activities with team development in mind. Many formal activities prescribed at the beginning of the team's life are designed to move the group dynamic toward one in which each member contributes effectively to the overall goal. As the members learn to work together and discover appropriate roles, informal, cooperative strategies may replace some of the formal processes. This chapter suggests some practical guidelines for evolving to a mature, effective team as quickly as possible.

BUILDING AN EFFECTIVE TEAM

At the beginning of a field project exercise, the project team consists of a group of acquaintances who know little about each other. Team members often are assigned to rather than volunteer for a project. Most team members do work on projects, but why do they bother? All the members have other things to do. The reason they work relates to perceived rewards for their performance. The perceived rewards may differ greatly from person to person. Most people want to graduate, or to keep their job and receive salary, or get a good grade. Other team members may obtain satisfaction or rewards from building a system that works or from building it rapidly, or from applying their skills to solve problems.

Start-up

When a team starts up, one of the shared objectives of the team members should be reaching mature performance as quickly as possible. Some of the activities

that facilitate growth and maturity, such as a team contract, may not appear to contribute directly to solving the client's problem. And the client may show little interest in how the team chooses to interact. However, team growth can contribute significantly to the overall success of a project.

The early stages of a project are the time when a team sets up initial roles and expectations about the people in the team and about how the team will function. The primary objective is to develop trust among members and trust in the team. The project usually starts with a meeting or a team-building exercise. In the early stages of the project, each team member should learn about the other team members and about what each can contribute. Some reasonable guidelines for the behavior of team members follow.

- *Facilitate instead of dominate.* A facilitator helps and encourages the other team members to express their views on the issues. A dominator aggressively provides an answer to every question that arises with little attention to the views of others. Only the most confident members of the team will speak up in the face of a domineering member. A team member is dominating if the member interrupts other people, insists that his or her view should always prevail, or tries to assign tasks to other people without their involvement in the decision.
- *Speak up.* In early meetings, team members form quick impressions about their colleagues that are hard to change. Members should talk about themselves and about what they can contribute. Even if members are not sure what they can do or how they can contribute, the team needs to know who they are. This initial period is trust-building time and team members trust (most of) the people they know more than they trust strangers.
- *Stay flexible.* Avoid locking into roles and answers early. Whatever plan develops at this stage will change and must change if the team is to grow. Establish the minimum structure needed at the beginning and add roles and assignments later.
- *Meet obligations.* A number of small tasks come up at the beginning of the project. Each team member should volunteer for one or more tasks, do it well, and complete it on time. Expectations are set early. If a team member only volunteers for later stage tasks, programming, for example, the team will not know what to expect until after the team chemistry has jelled. Similarly, good performance on the first few tasks builds confidence from other team members.

Team Evolution

A team may operate differently after it **matures** than when it began. At the beginning of the team-forming process, teammates may welcome such behaviors as taking charge and assigning roles. Once the team has matured, however, the same behaviors may create resentment. Similarly, such apparently destructive behaviors as confrontation and disagreement can help establish roles and define norms early in the team-building process but may create dissension in a mature team.

A well-functioning, mature team generally will make better decisions than any individual member. This improvement happens when the members use clues

from each other to identify good ideas; this utilizes the strength of each of the members. The members respect each other's opinions and everyone feels encouraged to contributing relevant information. Teams work together easily to accomplish such natural group tasks as brainstorming. Teams find working on such tasks as writing or programming more difficult. For effective teamwork, these tasks require project management—for example, task assignment and coordination. For five people to stand and watch one person write a paragraph, draw a diagram or code a procedure makes no sense.

Teams require time to mature. In the beginning, many teams defer to the loudest or most persistent voice. If the team remains in this stage, team performance may suffer and the members will not enjoy the experience. Effective teams move to cooperative work as quickly as possible. They provide meaningful roles for every member based on what each member does best. Not every team matures in a positive way. Some dysfunctional teams divide into factions, become dominated by a single individual, or come to resent freeloaders. Recognizing the potential of some of these behaviors can help the team to avoid problems and function effectively.

One organizational research model describes the maturing process with five stages (Greenberg and Baron, 2000, p. 256):

- **Forming:** establish ground rules and get acquainted (team contract).
- **Storming:** contention over control and leadership.
- **Norming:** establish relationships and come to an understanding of team expectations.
- **Performing:** work toward goals and getting the job done.
- **Adjourning:** disband the team.

In this representation, teams start with forming and storming, that is, testing each other, arguing, and trying to figure out roles. Organizations often try to facilitate this stage by scheduling retreats, special activities, or such exercises as preparing a team contract to build trust and understanding among members. Once the team gets through the forming and storming stages, the team can establish team expectations or norms. For example, the team may decide what process the team will use to reach a consensus. In the final stage, the now mature team works to get the job finished.

Teams often mature in a fairly abrupt way called "**punctuated-equilibrium**" (Greenberg and Baron, 2000, p. 257). According to this model, the team sets goals, defines tasks, and conducts a number of tasks described as forming, storming, and norming in the stage model. During this time, work proceeds in a very formal way, and progress toward goals may be structured and not very creative. Somewhere in the midpoint of the team's life, the members suddenly discover that they cannot meet their goals. At that point they begin to confront real issues more creatively, to reassign tasks, and to make the changes necessary to meet deadlines. The team then works in this new mode until they generate a last burst of energy necessary to finish the task.

TEAM CONTRACT

The **team contract** is a mechanism to facilitate team organization. The contract contains a written set of performance expectations for the team. A formal contract is most useful when team members come from a varied background and lack a common set of professional experiences—a typical situation for student teams. Many business teams omit the use of a formal contract because company policies, procedures, norms, and behavioral expectations predefine the behavioral expectations that might appear in a contract. Companies may use a formal contract when teams are made up of members from different branches of the company or from widely different geographical areas.

A team can be self-managed or directed. In business, many teams are directed: they have a leader or facilitator appointed by the organization whose job is to clarify the purpose of the team and to facilitate the members' contribution toward that goal. Typically the leader has status, knowledge, or experience that the other team members do not have. The leader may hold a management position in IT or in a functional area. The leader in a directed team also may have input into the performance review of the individual members of the team. In a self-managed team, the members often are roughly equal in status, experience, and knowledge.

In most field project situations, teams are self-managed. The team works out internal roles and responsibilities and establishes performance expectations. A self-managed team may define roles, responsibilities, and performance expectations in a formal written contract. Each member of the team may sign the document to indicate that he or she will abide by its terms. The content may include a statement of the team's purpose, an inventory of member skills, duties/roles of each team member, the team's code of conduct, and the leadership function. The statement of purpose gives the team a common goal or task. Normally the purpose of the team is to complete the project within the resources and time allocated.

Skills Inventory

Every project requires a range of skills, and each part or task of a project requires a distinct set of skills. Two issues arise. First, a well-functioning team will assign tasks or roles to the members with the most appropriate skills for the task. Second, one or more members of the team may need to learn new skills. Sometimes, the team may need to redefine the project scope to fit the skills the team possesses or can learn in the time available. Mastering a new skill within the project deadlines often poses serious problems. For all these reasons, the team should act as soon as possible to identify the skills possessed by the team, those skills needed for the project, and a process to acquire the additional skills, if any, needed for the project.

At the first meeting, the team can start identifying skills. When the members do not possess detailed information on each other's professional skills, conducting a **skills inventory** can provide insight. A sample skills inventory form is shown in Figure 2.1. Each member fills out the form. Using the information on the forms as a starting point, the team can then discuss the available skills of its members.

FIGURE 2.1 IT Project Skills Inventory

Area	Rank	Comments
<p>IT Project Skills Inventory Form Name: <i>Terrie Shaftkopf</i></p> <ul style="list-style-type: none"> • Instructions: For each skill, rank your expertise as one of the following: <ul style="list-style-type: none"> 5 in-depth knowledge and expertise or unusual ability in the area, 4 strong or above average knowledge and application skills or strong interest and willing to learn more as needed. 3 average knowledge, able to apply to problems, some experience with use, interest in learning more 2 know what it is, read about it in a text, some interest in learning more or 1 no knowledge and limited interest in learning about the area. • For the strong areas ranked 3 or above, note your background briefly in the Comments column: relevant courses, work or project experience in the area, etc. • List the resources—i.e., programs, computers, meeting place, etc. that you can contribute on the back of this form. 		
1) Technical Skills		
a) Programming		
i) Visual Basic	4	Completed VB course; use it
ii) Access	4	Use at work
iii) HTML	5	Maintain my company Web site
iv) Active server pages, etc.	3	Some experience at work
v) SQL	3	Database course
vi) Pseudo code	2	
b) Infrastructure		
i) Client/server	2	
ii) Telecommunications	2	
iii) Web	2	
iv) Database engines	2	
2) Analysis and Design		
a) Case tools	1	
b) ERDs	4	“A” grade in Database from Prof. Carte
c) Table schema	4	
d) DFDs	3	Systems course
e) Evaluation of alternatives	3	
f) Object-oriented design	2	
g) Other	—	
3) Writing and Checking Skills		
a) Writing skills	2	
b) Editing skills	1	
c) Checking work against the standards of the organization or manager	1	

4) Manager Skills		
a) Well organized	3	Manage my company Web site
b) Strong, good follow-up	3	
c) Project management experience or interest	1	
d) Good people skills	2	
5) Other—know and use MS Front Page	4	Use at work

Page 2

During discussion, each member can provide additional background information and describe his or her skills, including individual strengths and any resources he or she will bring to the team. With this discussion, the members get to know each other and can ask clarifying questions or make comments. For example, if a member lists Visual Basic (VB) as a skill, the team discussion may focus on refining how well the member knows VB. Or members may point out skills that a teammate forgot to list.

Using the information on the skills inventory forms and the discussion, the team may prepare a summary skills inventory for the team. If some needed skills are not available, the team can add skills acquisition to the project plan telling who will acquire the skills, how, and when. In addition to skills, the team requires other resources to carry out a project. Possible resources include a meeting place, computing facilities, contact with “experts”—people with skills the team does not possess but may need, ownership of or access to relevant software, and so forth. The team identifies what resources are needed and which resources are possessed as part of the skills inventory.

Assigning Roles

The next step is to allocate initial **roles**. Two critical issues drive the assignment of roles: (1) the work required in the project—the deliverables; and (2) the skills, abilities, and interests of the members. The team may wish to look closely at the first several deliverables before setting roles. The summary skills inventory and the discussion by the team will form a guide for determining the initial interests and abilities of the members. People perform better when they are asked to do things they are good at and want to do. The team may not find it possible or desirable to let everyone do exactly what he or she wants, but an effective team takes individual preferences into consideration.

The team should view initial roles as temporary for the first several deliverables or weeks. As the team matures and gains experience, the team will gain a better understanding of both the work needed for the project and the abilities of each team member. A good team reassigns roles as the understanding of the project and agreements among team members change.

A possible set of initial roles includes the following:

- *Coordinator*: Schedules meetings; keeps track of due dates for deliverables and team assignments; checks on and communicates progress, meeting dates, and places to all members. The coordinator serves as a facilitator, not a manager.
- *Communicator*: Takes charge of reporting and recording actions and products of the team. Fills out the weekly progress report to the manager and ensures that all members are informed of meeting dates and places. Sets up a Web site or other mechanism to allow every team member to view the latest versions of reports, drafts, and other products.
- *Standards manager*: Reviews all deliverables to ascertain that they meet the requirements set out by the organization. Maintains documentation including backup copies for all deliverables and other important items.
- Other initial roles might include *analyst, designer, writer, editor*, or one who takes primary responsibility, perhaps shared with one or two others, for the first deliverable or parts thereof.

Each member should have an initial assignment that allows the person to begin working on the assignment immediately. Team members should contribute throughout the project, not just in one period. As each week goes by, the team will gain a better understanding of the potential contributions of each member and the requirements of the project. This information will allow the team to reassign roles that better fit the members and the task.

Code of Conduct

The **code of conduct** describes the norms or ground rules the team members agree to follow. Teams should establish both task-oriented and people-related ground rules. In all team situations, disagreements and conflict often arise even with the most dedicated and well-meaning team members. The code of conduct provides a neutral forum for resolving conflicts without letting them become part of a personal dispute between two or more of the team members. The discussion can proceed in such terms as, “We all agreed to the following . . .”, instead of, “I object to what you are doing.”

Issues to consider when establishing ground rules include

- *Meetings*. How will the team set meeting times and places so that all members can attend? What do “on time” and “meeting attendance” mean? By what process and for what reasons can a member miss a meeting? What happens when a member misses a meeting without prior agreement or is late to one?
- *Roles and workload*. How will the team assign and/or reassign specific roles or tasks to members within each project phase? How will the team balance workloads among members?
- *Operations*. How will the team set deadlines and how firm will they be? How will the team make decisions, especially decisions that change the plan? How will team members give each other feedback? How will the team handle conflict or disagreements? What procedures will be used?

- *Evaluation.* What procedures will the team follow when a member does not meet his or her responsibilities? When the team conducts a peer evaluation, which behaviors of members will they reward and which ones will they punish?

The discussion for each rule should include such elements as

- The purpose of the rule.
- The specific statement of the rule.
- A definition of what constitutes a violation of the rule.
- The action(s) that occur when the rule is violated.
 - The procedure for taking action.
 - The person or persons responsible for taking action.
 - Exceptions, or mitigating circumstances that may lead to no action for a violation.

The team should state ground rules in a comprehensive and specific manner. A rule that says, “The team will set meeting times” does not follow the above guidelines and is too ambiguous to help the team function. A better statement might be similar to the one listed in the sample team contract in Figure 2.2. An actual team will want to create a more complete contract.

MANAGING A TEAM

Team management involves two functions, headship and leadership. **Headship** is the function of providing direction and assignments to the team. Headship provides organization and structure. **Leadership** is getting people to do what is desired. Leadership involves motivating, encouraging, and convincing people to get behind the success of the team. Illustrative functions of headship and leadership are listed in Table 2.1.

In self-managed teams, neither headship nor leadership nor any other roles are predefined. The team must work out appropriate roles for each member. Frequently a self-managed team will organize by selecting a person to serve as

TABLE 2.1
Headship and
Leadership
Functions

Headship Functions

- Makes assignments
- Communicates with management
- Keeps track of deadlines

Leadership Functions

- Demonstrates appreciation for everyone’s contribution
- Mediates conflict
- Promotes excellence
- Guides and motivates

FIGURE 2.2 Sample Team Contract

Team Contract

Team 7 GB Video Project

Team Purpose

Team 7 is assigned to the GB Video Project: design or select a new computer-based system to track and manage the rental and return of videotapes and DVDs as requested by the client.

Skills and Resources

The skills inventory summary for the team is attached as Appendix I. The team totaled the skills of each member to arrive at a measure of the aggregate level of skill in the team for each area.

The team has access to the following resources:

- An office, AH 405, where the team can meet and/or store materials
- Laptop computers: Al and Dan
- Desktop computers at home: Al, Terrie, and Dick
- Everyone has e-mail access from home
- Cars: Terrie and Dick

Initial Roles for Project Definition

- **Coordinator:** Terrie
Handles all contacts with the client and the manager. Schedules team meetings, manager meetings, and client visits. Keeps track of due dates for deliverables and team work assignments. Tracks progress against the plan. Prepares the agenda, questions, etc. for manager and client meetings.
- **Communicator:** Al
Reports and records all actions of the team. Works with the coordinator to communicate progress, meeting dates, and places to all members. Prepares the weekly progress report and e-mails it to the manager and all team members. Informs all members by e-mail of every meeting time/date and place. Sets up a Web site to allow every team member to view the latest versions of reports, drafts, and other products.
- **Editor and Standards Manager:** Dick
Completes and edits the draft of the SOW and the project definition report. Reviews all deliverables to ascertain that they meet the requirements set out by the organization. Maintains documentation including backup copies for all deliverables and other important items.
- **Analyst:** Dan
Writes up the information obtained from the client in the format, prepares the draft SOW, EDM, DFD and other project definition materials. Works closely with the editor.

The team will review and reassign roles when the first assignment is complete or as needed.

Code of Conduct

• Meeting Attendance

Purpose: To enable the team to conduct effective meetings and to make the best possible use of everyone’s time.

Statement: The team identified a set of meeting times (e-mailed to all members) that work for the team members at the initial organizational meeting. The team coordinator will notify each member by e-mail at least 24 hours in advance if one of the preestablished meeting times is not needed or if only some members are needed. Members must attend the meetings. The team will keep a written record of attendance at each meeting and post it on the team Web site.

Violation: Not showing up within 5 minutes after the set time or leaving before 1 hour (or the agreed-on end of the meeting if sooner) is a violation unless the member has notified the team by e-mail at least 8 hours in advance and received permission from the coordinator by e-mail for an absence. The coordinator or a designated representative will permit an absence for critical reasons—family emergency, severe illness, etc. The coordinator will report any excused absences to the team. If a majority of the team members feel that the excuse policy is being abused, they may require the coordinator to get approval from a majority before granting an excused absence.

Action: The coordinator will notify the person by e-mail that they missed a meeting. Each missed meeting deducts 1 point from the individual’s team score. Three unexcused misses will result in a 5-point deduction. Members will consider the team score when filling out peer evaluations.

-
-
-
-

- Changes. By majority vote, the team may grant exceptions to or change any of the rules at any time. Fairness, effectiveness, and consistency shall form primary tests for all actions.

Signatures

Dick Von Kemp

Al Price

Terrie Shaftkopf

Dan Cartperson

Appendix I. Summary Skills Inventory for Team 7.

Skill Areas	Team Members				Total
	Terrie	Dan	Dick	Al	
1) Technical Skills					
a) Programming					
i) Visual Basic	4	5	1	2	12
ii) Access	4	5	1	3	13
iii) HTML	5	4	1	3	13
iv) Active server pages, etc.	3	3	1	3	10
v) SQL	3	5	1	3	12
vi) Pseudo code	2	3	1	3	9
b) Infrastructure					
i) Client/server	2	4	1	3	10
ii) Telecommunications	2	4	1	3	10
iii) Web	2	4	1	3	10
iv) Database engines	2	5	1	3	11
2) Analysis and Design					
a) Case tools	1	3	3	3	10
b) ERDs	4	5	3	3	15
c) Table schema	4	5	3	3	15
d) DFDs	3	5	3	3	14
e) Evaluation of alternatives	3	3	3	3	12
f) Object-oriented design	2	2	4	3	11
g) Other	—				0
3) Writing and Checking Skills					
a) Writing skills	2	3	4	3	12
b) Editing skills	1	2	4	3	10
c) Checking work against the standards of the organization or manager	1	4	5	3	13
4) Manager Skills					
a) Well organized	3	3	4	3	13
b) Strong, good follow-up	3	3	4	3	13
c) Project management experience or interest	1	4	4	3	12
d) Good people skills	2	4	4	3	13
5) Other—know and use MS Front Page	4	4	1	3	12

coordinator—a person to provide some aspects of headship. The coordinator also may provide some leadership. But often leadership and headship come from several members and vary over time. In a well-functioning team, the roles, including the coordinator role, will change as the team members come to understand how the relative strengths of the members match the task at hand. Flexibility and rapid response to problems or changes are strengths of the self-managed approach.

Most of the discussion in this chapter assumes that project teams are self-managed because most class project teams are self-managed. In organizations, many teams are directed. In directed teams, the organization assigns a person to take charge of the team when the team is initially formed. Common titles for the role include **team lead, project director, or project manager**. Usually the assigned director has experience, knowledge, and/or skills that make him or her qualified to guide the actions of the team and to evaluate the performance of the other members of the team. The team director starts out with responsibility for the headship and leadership functions of team management, including the role of coordinator. He or she then may delegate some of the headship and leadership functions to other team members on occasion.

In many cases, the assigned head is not the only or even the major leader on the team. Other team members may earn some of the leadership functions on the basis of their performance. Individual members often have a very large role in influencing the behavior of other team members and in facilitating the success of the project. Leadership becomes particularly important with teams composed of members from both the client and the IT community. In many of these teams, the formal project manager comes from the client community to assure alignment with organizational requirements. But a client manager may lack the technological credibility and cultural experience to lead or motivate the IT members of the team. These teams usually have a senior IT member who is not the project manager, but who is expected to provide leadership among the IT members in support of the project goals.

Successful Teams

The most successful teams consist of people with a variety of personal skills and technical abilities who are willing to work for the team. The management task is to provide a team environment that makes people willing to work for the team. Reflect for a moment on the old saying, “Everyone does at any moment the thing they most want to do.” This saying is a tautology, a truth by definition. In a project environment, the saying implies that the team needs to create an environment in which the team members want (or at least are willing) to do what the project needs.

Many people hold the misconception that team member motivation works mostly or only on the basis of salary or grade rewards and firing or failing threats. These kinds of rewards and punishments clearly produce some effects, but relating significant short-term changes in salaries or grades to performance on a specific project task is difficult both in companies and universities. A team member might reasonably expect to get a B (or A or C) regardless of how well they do a specific task. Often, field project team members respond to other interpersonal

rewards much more readily than to grades. The rewards may include learning new and desired skills, personal satisfaction for good work, and peer approval.

Individual Needs

The art of managing a team for success, both for self-managed and directed teams, relies in part on providing team members the rewards they want from the experience. The “art” part of that statement comes from the fact that different people want different things, that is, they value different things. The Trichotomy of Needs model offers a useful behavioral framework to identify what people value (McClelland, 1961). McClelland uses the motivational theories of contemporary psychology to extract three basic needs that people value when making decisions about what to do.

McClelland observes that most people are motivated by some combination of the following three needs:

1. A need to achieve (nAch).
2. A need for peer acceptance and affiliation (nAff).
3. A need for influence or power (nPow).

This motivational framework can help the team to identify the kind of rewards that each member will value. The model at best provides a starting point, not an exact prediction of how anyone will behave. The concepts in the model are approximations and generalizations. Actual team members may or may not behave as the model suggests and may behave differently from one situation to the next. But using the model to characterize more clearly the preferences of each member can help the team to organize and function in ways that reinforce motivation for every member.

The **need for achievement (nAch)** represents a need to accomplish something or to attain a goal. People with a need for achievement are more concerned with the accomplishment than with a reward. They focus on the outcome and tend to disregard the process rules for getting there. They feel a need to contribute and want recognition for their individual contribution. They like accurate feedback from knowledgeable evaluators on their work, either positive or negative. They tend to make optimistic estimates of what they can do until they get information, but they don’t like real risk.

Most IT programs start people in a computer programming class that forces an achievement-oriented approach. People who don’t like this experience are not attracted to the MIT field. A team should expect several of its team members to have a high need for achievement. Indeed the best technical members are likely to be the most motivated by achievement.

From a team point of view, expect nAch members to tend to

- *Demand to be included in decisions.* If they are not included, they will probably not work to support the team. They tend to be very bad at routine work. They have a hard time letting go of their ideas.
- *Try to take on a whole task.* (nAch) people want to own a piece of the project themselves and may not want to collaborate.

- *Avoid conflict.* High-achievement people are disturbed by conflict and will often give in or withdraw from participation if there is much team contention.
- *Undervalue interpersonal relationships.* Achievement people may want to get directly to the task and not recognize unsaid concerns or wounded feelings in team members. They can have difficulty with a leadership role.

The **need for affiliation (nAff)** represents a need for group acceptance and approval. These people participate because of the group rather than use the group to get something done. High-affiliation people are willing to do whatever tasks the group needs. They prefer to work collaboratively rather than alone and to value social time, even if it is not productive. They are uncomfortable taking stands that might alienate someone in the group and have a hard time setting goals.

Expect nAff members to tend to

- *Be good staff support.* Affiliation people are willing to do partial tasks and support other people's efforts, even if they don't get credit for it.
- *Expect good social relationships.* High nAff people enjoy parties and may introduce purely social topics into group meetings. Their feelings can be easily hurt by negative feedback or conflict.
- *Have difficulty defining and completing tasks.* This dysfunction is particularly true if one person alone assigns the tasks for independent work.
- *Be uncomfortable as leaders.* While affiliation people will work to make a group become cohesive, they have a hard time demanding performance from others.

The **need for power or influence (nPow)** is a desire to accomplish a goal through others rather than directly. The nPow motivation can exist as either socialized or unsocialized power. Unsocialized power is used for the good of the individual regardless of the consequences to others. The exercise of unsocialized power leads to bullying behavior, "using" people, and team resentment. Socialized power is used to further the goals of the group. High-power people make good leaders and enjoy making decisions for others. They understand process as a tool of control. They enjoy conflict and emphasize winning. They accept risk and like high-visibility rewards.

Most corporate managers have a high-power motivation. Recognizing this nPow orientation can help the team to work well with executive clients. IT people, on the other hand, tend to be uncomfortable with power positions and nPow people.

Expect nPow people to tend to

- *Enjoy leadership.* People with a high-power motivation will volunteer to lead and probably try to organize the group. They may try to take over the group and ignore contributions from other members.
- *Prefer representing to doing.* People with an nPow motivation are usually comfortable with public speaking and make very good presenters and front people. They are less comfortable working without recognition or support.
- *Introduce conflict.* High-power people view conflict as a legitimate way of settling disagreements, and they do not hold grudges about battles they lost. They don't understand people who back down or resent confrontation.

- *Focus on process.* Rules and procedures are used to manage the behavior of the group. NPow people are uncomfortable with skipping steps or circumventing people in charge.

Effective teams contain a mix of people with different motivational drives. Each mix brings potential advantages and dangers. Two strong-willed nAch people, however, might compete over the “best” way to do a project unless someone else mediates. Two nPow people may disrupt a meeting until one wins and the other withdraws. When teams recognize the kinds of disruptive behavior that arise from motivational mismatches, they will be better prepared to mediate.

Leadership from a Motivational Perspective

As noted, organizations possess only a limited ability to grant or deny tangible rewards for day-by-day behavior. Raises usually come only once or twice a year. Awards normally are handed out at the end of projects or at special events. Offices and assignments change infrequently. In this environment, leaders and managers need to use intangible rewards for ongoing motivation and control. The McClelland model, in contrast to most other behavioral profiles, helps to identify the intangibles that motivate people. Understanding that different people work for different internal reasons helps a leader to replace grudging acceptance of assignments and tasks with willing participation.

Within a team, a leader needs to identify individual motivating desires and use them to determine “payments” for supportive behavior. For example, having team members compete for a trophy or prize can provide strong motivation for an nPow member, but it may decrease motivation for members with nAch or nAff profiles. However, putting an nPow member who has done nothing on the project in charge of the final presentation because it is “the only thing he can do” may appear to reward poor behavior. To insist that the nAff member conduct the final presentation because she has attended every meeting, provided help whenever she could, and generally made the project a pleasant experience may upset a good performer. Programmers, often nAch people, want accurate feedback from knowledgeable people rather than unearned or generalized praise. The nAch programmer who says, “Let me code the entire Web interface; I don’t really need any help” probably means it. When the programmer completes the task, praise from the most senior person around will be appreciated.

The process of developing the tools for motivational leadership starts with trying to identify the motivational profile of different members of the team from their behavioral clues. Given the high likelihood of missing information and ambiguity, leaders should be prepared to revise their assumptions as a project goes on. People provide clues in their conversation, in their office appearance, in their performance, and in their reaction to different situations that they encounter. Once a leader has formed an opinion of the motivational profile of a team member, the leader can make good guesses about what rewards that person would value. One option is to ask, “What do you think is a good way to motivate X?” Many people will respond with what would motivate them, provided they identify with X. The goal of good motivational leadership is to find and deliver positive,

meaningful rewards for desired performance. Leadership is an art and motivational analysis is merely a tool to support that art. The leader tries to understand both explicit and implicit preferences of people in order to make the team experience as rewarding as possible for both the organization and the team members.

WORKING IN A TEAM

Working in a team setting offers advantages but also poses difficulties. If the team tries to do most of the work in meetings as a team, the meetings consume a lot of time and may frustrate some or most of the members. If much of the work is done individually or in subgroup meetings, parts of the project may end up undone, incorrect, or duplicated unless the team places extra effort on organizing, managing, and communicating. Project management, an important topic, is discussed in Chapter 3.

Communication

Most teams depend on communication between members to function effectively. As each person develops assigned parts of the project, he or she may need to coordinate with others working on related parts. All team members require communication to learn about progress and any changes in deadlines, meetings, plans, and standards or conventions. An effective team will focus communication on assignments and progress not on speculation about the motives or personal lives of the absent team members. Inevitably, both kinds of content appear, but the balance should go strongly toward the constructive sharing of information over dysfunctional gossip.

The major mechanisms for communication are face-to-face discussions in teams or subgroups, telephone conversations, and sharing written documents and electronic communications. Team meetings can provide good communication. These face-to-face sessions early in the project can develop team cohesion and trust. As the project proceeds, freeform meetings, where everyone just comes and free-associates, can generate some progress, but they mostly waste time. Communication effectiveness is enhanced by an agenda where members report briefly on issues and progress, and the team members respond with suggestions and comments. The coordinator or other members can develop and distribute an agenda and then follow up to see that people are prepared for the meeting. The team should develop an agenda for the next meeting during each meeting. At a minimum, the team can spend the first few minutes of an unplanned meeting developing an agenda. Handouts distributed before the meeting and read in advance by the members always help.

Electronic communications can greatly facilitate teamwork. The simplest form consists of e-mail. E-mail can save a lot of time and communicate with less ambiguity. Meeting announcements or changes in plans fit well into e-mail. For e-mail to work, the team members must check their e-mail regularly. An unread message provides no communication. (In this respect, nothing beats a telephone for a last-minute message if the person happens to answer.) E-mail also works well for the distribution of draft documents or reports and for feedback.

Managers and some clients prefer e-mail communication for routine matters. Every team should take full advantage of e-mail.

A second step in utilizing electronic communications fully is for the team, early in the project, to agree on a location where the current version of every project component is stored. The location can be a common data storage drive available to everyone, a managed listserv location and/or other Web site. The Web sites are good because members can access them from home, work, or travel. The team should appoint a location coordinator to assure that the contents are current and complete. The team also needs either formal or agreed-upon access control rules: these determine who can add items to the site, change them, and/or delete them.

Electronic communication sometimes causes problems because of disinhibition. Team members, as a result of years of socializing, develop standards of acceptable behavior for face-to-face, written, and telephone communication. Sometimes members do not apply these same standards to such newer media as electronic communication. Standards for effective electronic communication (as well as all communication) include the following:

- *Avoid flaming.* If a team member is mad or upset, he or she should calm down before communicating. Sending hostile or derogatory messages to all or part of the team will not lead to constructive communication. Maintain professional standards in all team communications.
- *Sign and date communications.* Including versions, dates, and the author's explanatory comments in the electronic header of all documents helps the coordinator and all the members to avoid replacing a new document with a previous version. When the final product is assembled, the editor can remove the markers.
- *Assign ownership or access control to each document.* Only the owner can authorize changes to the official copy of the document. Others working on the same component can submit suggested changes to the owner. Allowing everyone to change the official copy leads to chaos and discord.
- *Always maintain a backup copy of everything important at a separate, secure location.* A manager will show little sympathy when a team tells him or her that the machine ate the report.
- *Copy all members.* When the team distributes versions of a product, send or make it available to every team member. If the team has posted a major change to a Web or listserv location, e-mail the new version or notice thereof to every team member.

Manager Relations

While self-managed teams establish their own internal goals and processes, the team's client and manager set the external expectations for the team. The client, or perhaps the sponsor, is the person responsible for setting the organizational expectations for the project and for determining acceptable outcomes. The manager is responsible for the quality, completeness, and timing of the work, obtaining resources for the team and for personnel evaluation (i.e., grades). The team bears major responsibility for behaving professionally toward the manager and for developing an effective working relationship with the manager.

A team in an organization may report to a single manager or, perhaps, to several, for example, an IT manager and a client manager. A committee also can serve as the manager for a team. The manager may assign projects and members to teams, track performance, and provide a final evaluation. The manager also may offer or agree to help with technical and political problems for the project. The manager shares responsibility for the success of the project with the team. Most managers perform other duties and can spend only part, perhaps a small part, of their time with any one team. In a field project exercise, an instructor usually acts as the manager for the team.

Some guidelines for teams to use when interacting with managers include

- *Keep the team manager informed of progress.* Most important, warn the manager as promptly as possible of any difficulties. Managers dislike problems, but they hate unpleasant surprises. Managers cannot help solve or adjust for a problem until they know the problem exists. They tend to punish the team for problems that are revealed too late for corrective action or are revealed by someone outside the team.
- *Respect the manager's time.* Most managers are busy. Meet with the team manager on a regular basis to review work and discuss difficulties. Take the initiative to schedule or reschedule meetings; do not make the manager come looking for the team members. Make appointments for meetings; do not drop in casually unless team members are certain the manager encourages such visits.
- *Ask for help when needed.* The manager will help directly with some of the problems and will help the team find needed resources. Team members should do their part first, however. Try to solve the team's problem before bringing it to the manager.
- *Manage client relationships.* A manager should never learn about a problem or a development from the client. He or she should hear it from the team first. Before scheduling meetings for formal presentations or conferences, check the availability of the manager. Do not present the manager with a date on which the team and the client have already agreed.

DEALING WITH NONPERFORMING MEMBERS

Teams, especially self-managed ones, can encounter problems with some members. Team members are expected to contribute their best work to complete the project. Most members contribute well if the team pays attention to the needs of all of its members. Every member of the team has the responsibility to make sure that all the members do their part and contribute to the welfare of the team.

Team members who are disruptive—difficult, domineering, or dogmatic—or ones who are unable or unwilling to carry their share of the work can generate resentment and resistance, and can reduce team productivity. If one or more members fail to meet their responsibilities, the team may believe it faces three choices: (1) convince the errant member to change; (2) expel the errant member; or (3) live with the problem. Alternatively, if a member feels abused or unable to work with the others, the member may wish to resign from the team.

The best way to deal with any problem is to solve it as rapidly as possible in a satisfactory way. When a majority of teammates decide that a member or member(s) are not performing at a satisfactory level, appropriate steps include

1. Review and encourage everyone to follow the team contract.
2. If the problem persists, try to resolve it in a low-key way that does not create embarrassment or angry divisions within the team. For example, select a person to talk privately with the errant member to make sure he or she understands what the team expects. Ask the errant member about any problems or issues inside or outside the team that may be affecting his or her performance.
3. If the problem still persists, ask the team manager to talk with the person.
4. If no improvement occurs, the team as a team informs the person that their performance is unacceptable and that if no improvement occurs, the team will consider further actions.

Hopefully, the errant member will respond to the information and to peer and manager pressure to perform better when informed of expectations and/or problems. If no improvement occurs, the team may consider the next step, removal. Timing may affect the appropriate action. If a member refuses to perform at the beginning of a project, removal may represent a good option. But if a member refuses to perform the week before the final report is due and resists all applications of peer pressure, the other team members have little choice but to shoulder the additional load. The members can express their displeasure in the peer evaluation process. The overriding goal is a successful project outcome, whatever it takes.

Removing a Member

Occasions may arise when one or more members prove unable or unwilling to contribute a fair share even after individual, team, and manager counseling. The **removal** of a team member is an extreme option and a rare one. A member who is removed may face severe penalties. A team member facing removal is wise to take every possible step to correct the situation. But when removal is deemed necessary, the team, with the consent and participation of the manager, can use a multistage process. Each organization has its own policy for removal to assure fairness and to protect against lawsuits, but most work by notifying the person of his or her unsatisfactory performance, specifying what must be done to correct the performance and then measuring outcomes.

A possible process to dismiss a member from a team might proceed as follows. The team first consults with the manager about the problem before taking any steps toward removal. The manager normally reviews the team's grievances and efforts to resolve them, then may wish to meet with the entire team, may ask the team to make an additional attempt to resolve the problem, and may wish to meet privately with the offending member.

If the manager agrees that a problem remains and further action is warranted, the team can put the member "on plan." Possible steps for doing this include:

- Convene a team meeting at which the team decides what the errant member must do to achieve acceptable performance, that is, meeting the plan terms.

The team should define the desired actions, due dates, and quality standard clearly in writing and use objective measures. The performance specified by the plan should be similar to what is expected from other team members.

- Notify the team member in writing that his or her performance is unsatisfactory using an objective statement approved by the manager. Provide the member with a copy in writing of the proposed plan.
- The errant member can agree in writing to the plan. He or she should could agree to meet the plan conditions by signing the plan or sending written notice of agreement to all members and the manager. If the member refuses to agree to the conditions, the team probably will choose to remove the member from the team.

If the team subsequently believes that the member meets all the conditions of the plan, the team takes no further action. If the team concludes that the member did not substantially fulfill the plan, the team can document the shortfall and present it to the manager. The manager normally makes the final decision on removal. People removed from a team for unsatisfactory performance generally receive sanctions from the manager, such as they are ineligible for a raise, they receive a grade reduction, or such. In actual organizations, the person's employment sometimes is terminated, but often the person is just assigned to another task.

Resignation

Resignation from a team represents another extreme and disruptive option. A member may risk sanctions when he or she asks to resign from a team. A wise team member takes steps to avoid problems prior to or at the time of team formation. When an existing and severe interpersonal conflict exists between potential team members, such as long-time antagonism or a failed romantic relationship, the members should bring the issue to the attention of the manager at the beginning of the project and request a new assignment at that time. Once the project is underway, a request to resign identifies the team member as a person who cannot get along.

A member should make every reasonable attempt to resolve differences before asking to resign. Members are expected to set personal conflicts aside at work and to interact on a professional basis. Members also are expected to work through their conflicts and disagreements. Requests to resign from a project because the team is not following a member's suggestion or because the other members' work habits are not compatible are seldom viewed favorably. However, circumstances exist when a work situation, if allowed to continue, may cause serious damage to members or to the deliverables. In this case, ask the team manager to discuss the merits of a resignation.

Dysfunctional Teams

Sometimes the problems faced by a team come not from a nonperforming member but from the interactions between several or all of the members. Some of the more common team interaction problems are discussed below.

Fragmentation occurs when two team members, often nPow types, insist on following incompatible approaches. Sometimes the differences result from differing

views of an appropriate approach, but often the split indicates a contest for recognition and control among the members. In any case, the infighting consumes most of the energy in the team, and the nAff and nAch members either join subteams or withdraw. The nonfighting members need to stay together and take a role in resolution. If the team contract contains a dispute mechanism, one of the team members needs to invoke it. If not, the other members can help by describing the problem openly as a conflict and staying focused on resolution, rather than on criticizing the fighters. In the absence of any progress, the team manager can mediate the problem by calling a team meeting. The wise manager seeks to facilitate a solution by the team rather than impose one on the team. The “loser” in these contests may not provide much further contribution to the team.

A **hijacked team** occurs when a single, strong-willed person imposes his or her vision on the team and ignores and suppresses contributions from other members. Frequently the hijacker asserts his or her solution and takes over most of the work at the beginning of the project. In this situation, the other members, without thinking about the consequences, may disengage. Some members may defer to the hijacker with the attitude, “If she wants to do all the work, let her.” Others may become discouraged with the rejection of their input and just quit. Note that the problem involves two components: (1) the hijacker and (2) the passivity of the other members. The highjacker cannot succeed without the implicit consent of the highjackees.

The hijacked team faces two serious problems: (1) the team loses many good and often the best ideas; and (2) more importantly, when the workload increases beyond what the hijacker can do, the other members of the team may decline to reengage. The project is no longer their project. The situation often results in a poor project outcome to the detriment of all of the team members.

The best solution is to confront the hijacker early before the other members disengage. The other members must insist that contributions from all team members receive fair consideration and that the team share the workload. If the highjacker continues to defy the rest of the team, one or several members need to involve the team manager. The manager can reemphasize the basic tenet of a team project: all of the team members share the responsibility for and consequences of the product. The manager also can counsel the hijacker individually about effective team participation. In the worst cases, the manager can encourage the team to put the highjacker on plan and, if necessary, proceed to remove the highjacker from the team.

The **secretive genius** problem occurs when a member of the team tries to assert exclusive, confidential ownership of some part of the solution. Often a programmer wants to develop the code by himself or herself and not share it with the team until it is “complete,” but the same problem can happen with other parts of the project. Unfortunately, this behavior may defer review of the “genius’s” product until the genius is so invested in the work that he or she fights all change and improvement. In any event, this behavior imposes an unnecessary burden on the team members who may need information about the genius’s work in order to proceed with their part of the project and reduces the lead time for the team if corrective action is needed. Sometimes the genius will produce a poor product or none at all, but more often the genius produces a

satisfactory product at a time that is too late to correct any problems and to coordinate this part of the project with everyone else.

The team can address this situation by holding one or more formal walk-throughs for the final report (see Chapter 13). At this walk-through, the team must insist that the genius fully describe his or her progress and results. By covering all parts of the team's work, the walk-through will look more like a constructive review and less like a vendetta against the genius. If some members of the team are concerned about participation by the genius, the team can ask the manager to attend the walk-through. Most often, the manager's presence convinces the genius to participate and, if not, the manager has clear evidence of the problem. The manager can then insist that the genius make a full presentation of his or her work at another walk-through in a few days.

PEER EVALUATIONS

Peer evaluations offer an opportunity for team members to comment on the performance of their peers. For example, the team may ask its members at a midpoint in the project to self-evaluate their own performance and to note any suggestions for improving team effectiveness. The goal is to provide information during the project that will allow the participants to modify their behavior for the success of the project. Many actual organizations use a step midway through the project called a 360-degree review. In these reviews, team members are asked to evaluate their peers, managers, and subordinates on a collection of specific traits and behaviors that are important to the project at hand. These evaluations have both strengths and weaknesses. Although they simply may validate popularity, when filled out thoughtfully, they can give each person useful feedback.

The prospect of a "good end of project evaluation" may serve as a reward to motivate some team members during the project. At the end of the project peer review, each person is asked to evaluate the total or overall performance of his or her team members during the complete project. A sample end-of-project peer review form is depicted in Figure 2.3. When team members fill out a peer review, they should remember that the focus is on only those issues that affect job performance. Each person should answer the items based on actual contributions made, not on personality or preferences for friends. When a team member gives one or more teammates credit for exceptional performance, the "fixed-sum" scoring scheme described in Figure 2.3 requires the member to reduce the credit for someone else. When some members perform better than others, the team should give the better performers better-than-average scores to preserve the credibility of the peer evaluation process.

Team members value and expect fairness. When members do a good job, they want to be recognized for their work by their team, manager, client, and the organization. They might hope for a higher salary, a good grade, a new choice assignment, or praise. Fairness in a field project poses many problems. The manager tries to evaluate the deliverables and the team members. Some projects are more difficult and require more effort, but the manager can estimate relative effort and

FIGURE 2.3 End-of-Project Peer Evaluation Form

Peer Evaluation

Name _____ Team _____

You can allocate a total of 10 points times the number of team members other than yourself—i.e., if the team has 6 members, allocate 50 points; if the team has 7 members, allocate 60 points. Assign from a maximum of 15 to a minimum of 0 points to each member to recognize his or her contributions to the team’s performance. The team may give every member 10 points. However, if one or more people made above-average contributions, they deserve a higher rating than 10. Ask the team manager if anything is unclear.

List all the team’s members except yourself in alphabetical order:	Score (0 to 15)
1. _____	_____
2. _____	_____
3. _____	_____
4. _____	_____
5. _____	_____
6. _____	_____
7. _____	_____
The Total must equal the points you have to allocate.	_____
Brief reason for lowest score:	
Brief reason for highest score:	

difficulty poorly, if at all. The manager tends to look for such things as the timeliness and quality of the deliverables, evidence of innovation, excellence in applying knowledge, and adherence to standards. Hopefully the manager can obtain input from the client to augment his or her own observations. The manager also receives weekly or periodic progress reports from the team.

Summary

In a modern organization, analysts and designers undertake many projects as a member of a team. A project *team* consists of two or more people jointly responsible for accomplishing work on a project: the goal is a “successful” project. Teams may reduce the time needed to complete a project and may improve quality by bringing a wider range of experience and providing a potential way to best use the strengths of each team member. Intelligence, technical and business knowledge, and willingness to work hard all contribute to becoming an effective team member but success also requires an understanding of how teams function.

Team characteristics for IT projects are as follows:

- A team contains two or more *members*. A manager generally assigns members to the team.
- Team members share a common purpose or goal, which is to create a satisfactory IT product or system for the client within the constraints.
- Team members possess different skills. A well-functioning team makes assignments to take advantage of the best skills of each member.
- Team members are mutually accountable for the success of the project.
- Organization and management provide mechanisms to improve the effectiveness of a team.
- Team members need to communicate. Mechanisms and technologies that facilitate communication offer tools to increase effectiveness of the team.

When a team forms, one of the shared objectives of the team members should focus on reaching mature performance as quickly as possible. During the early stages of a project a team sets up initial roles and expectations about the people in the team and about how the team will function. The primary objective is to develop trust among members and trust in the team. One organizational research model describes the maturing process with five stages.

1. *Forming*: establish ground rules and get acquainted (team contract).
2. *Storming*: contention over control and leadership.
3. *Norming*: establish relationships and come to an understanding of group expectations.
4. *Performing*: work toward goals and finishing the job.
5. *Adjourning*: disband the group.

In most field project situations, the teams are self-managed. The team works out internal roles and responsibilities and establishes performance expectations. A self-managed team may define roles, responsibilities, and performance expectations in

a formal, written team contract. The content can include a statement of the team's purpose, an inventory of member skills, the duties and roles of each team member, the team's code of conduct, and the leadership function. The statement of purpose gives the team a common goal or task. The code of conduct sets forth the rules under which the team will operate. The code provides a neutral forum for resolving issues without letting them become part of a personal dispute between team members.

The most successful teams consist of people with a variety of personal skills and technical abilities who are willing to work for the team. The management task is to provide a team environment that makes people "willing to work for the team." Team management involves two functions, headship and leadership. Headship is the function of providing direction and assignments to the group. Headship provides organization and structure. Leadership is getting people to do what is desired. Leadership involves motivating, encouraging, and convincing people to get behind the success of the group.

The art of team management, both for self-managed and directed teams, is to provide team members the rewards they want from the experience. The "art" part of that statement comes from the fact that different people want different things, that is, they value different things. The Trichotomy of Needs model offers a framework to identify what people value (McClelland, 1961). McClelland observes that most people are motivated by some combination of the following three needs:

1. A need to achieve (nAch).
2. A need for peer acceptance and affiliation (nAff).
3. A need for influence or power (nPow).

At best the model provides a starting point, not an exact prediction of how anyone will behave. Actual team members may or may not behave as the model suggests.

Most teams depend on communication between members to function effectively. As each person develops assigned parts of the project, he or she may need to coordinate with others working on related parts. All team members require communication to learn about progress and any changes in deadlines, meetings, plans, and standards or conventions. The major mechanisms for communication are face-to-face communication in teams or subgroups, telephone communications, written documents, and electronic communications including e-mail and Web postings.

Team members are expected to contribute their best work to complete their project. Most members contribute well if the team pays attention to the needs of all of the members; however, teams, especially self-managed ones, can encounter problems with members. When one or more members fail to meet their responsibilities, the team needs a process that may include removing the member from the team. Alternatively, if a member feels abused or unable to work with the others, the member may wish to resign from the team.

Peer evaluations offer an opportunity for team members to comment on the performance of their peers. For example, the team may ask its members to evaluate their own performance and to note any suggestions for improving team effectiveness at the midpoint of the project. The goal is to provide information during the project that will allow the participants to modify their behavior for

the good of the project. The prospect of a “good end-of-project evaluation” may serve as a reward to motivate some team members during the project. An end-of-project peer review asks each person to evaluate the total or overall performance of his or her team members during the complete project.

Key Terms	adjourning, 38 code of conduct, 42 communication, 36 effectiveness, 36 forming, 38 fragmentation, 55 goal, 35 headship, 43 hijacked team, 56 leadership, 43 matures, 37	members, 35 mutual accountability, 36 need for achievement (nAch), 48 need for affiliation (nAff), 49 need for power (nPow), 49 norming, 38 peer evaluations, 57 performing, 38 project director, 47	project manager, 47 punctuated-equilibrium, 38 removal, 54 resignation, 55 roles, 41 secretive genius, 56 skills inventory, 39 storming, 38 team, 33 team contract, 39 team lead, 47
------------------	---	--	--

Review Questions

1. What is a team?
2. What are the purposes of establishing a team contract?
3. What are the major areas or issues that a team should include on a team contract?
4. What is a reasonable method of taking an inventory of skills and why is a skills inventory important?
5. What are some of the issues a team should consider when establishing ground rules to follow as a team during the project?
6. What responses should be made when a team ground rule is violated?
7. Describe and define the team five-stage maturity model.
8. For what reasons might a team member work on a team project?
9. Describe the characteristics of a person with a need for achievement.
10. Describe the characteristics of a person with a need for influence or power.
11. Describe peer evaluation.

Critical Thinking Exercises

Individual Exercises

1. Set up a plan for communicating with your team during the project. The plan should include assignment and reporting mechanisms.
2. Prepare a skills inventory for yourself.
3. Prepare a team contract for a team that consists of you plus a roommate, spouse, or parents.

Group Exercises

1. A team member with a 4.0 GPA at a university is a sergeant in the armed forces. He attends the university on assignment and will receive a commission as an officer upon graduation. He becomes irritated when he perceives that other members on his team do not work as hard as he expects. He frequently chides his fellow team members to

work up to their full potential. He is very strong willed and gets upset if the team does not accept his ideas.

- a. What type of individual is the sergeant?
 - b. What suggestions do you have for the team to achieve a better working relationship?
 - c. What suggestions would you have for the sergeant to better understand the team and its goals?
2. A team consists of members who know each other well from working on other assignments. They look forward to working with each other and believe that they will get along well. They completed a skills inventory as part of the team contract. All the students have good programming, communication, and analytical skills. During their first walk-through with the client, they discover they need ASP. Dick, one of the team members, knows ASP and had used it at another assignment. The team immediately assigns Dick to the programming chores. Other team members are assigned to the graphical and writing responsibilities. The team believes that since Dick is going to code, he does not have to work on other aspects of the project. As the project progresses, Dick becomes upset with the team and complains that they are not doing their part. When nothing happens, he begins to skip meetings and also misses the deadline for the ASP program. The project fails and all the team members receive penalties.
- a. What should have happened when the team discovered the need for ASP?
 - b. During the course of the project, what could have been done to prevent the problems?
 - c. According to the McClelland model, what type of individual is Dick?
 - d. Do you think this is a rare incident?
 - e. Is it unusual for a team to need a new skill?
3. Judy is a very dynamic person on a team with two other women and three men. Team member Bert seems to really enjoy Judy's company. As the project progresses, Judy and Bert begin to see each other socially. Other team members become concerned about Judy's and Bert's "shoddy work." Judy and Bert respond slowly if at all to work-related e-mail and express little interest in the project.
- a. Should the team contract discourage this type of behavior?
 - b. What should the team members do about this situation?
 - c. How and who should talk to Judy and Bert?

References

- Greenberg, Jerald, and Robert A. Baron. *Behavior in Organizations: Understanding and Managing the Human Side of Work*. 7th ed. Upper Saddle River, NJ: Prentice Hall, 2000.
- McClelland, David C. *The Achieving Society*. New York: Van Nostrand Reinhold, 1961.

Chapter Three

Project Management

Chapter outline

Introduction

Project Planning

- Using the SDLC for Planning*
- The Spiral Model for Project Planning*
- Flexible Project Planning*
- Planning Mechanisms*
 - Client/Team Contract Plans*
 - Planning with Joint Teams*
 - Prototype-Based Plans*
 - Outsourcing Plans*
- Generating the Plan*
 - Selecting the Activities or Tasks*
 - Task Times and Sequence*
 - Constructing the Schedule*

Statement of Work

Project Execution and Control

- Project Execution*
- Controlling Changes in Operations*
- Controlling Changes in Requirements*
- Monitoring Progress Against the Plan*
- Taking Corrective Action*
- Project Review Points*

Project Management Tools

Project Communication

- Progress Reports*
- Written Reports*
 - Table of Contents*
 - Headings and Fonts*
 - Executive Summary*
 - Introduction*
 - Good Writing*
 - Report Appearance*
- Presentations*
 - Team Member Roles at the Presentation*
 - Visual Aids*
 - Rehearsal*
 - For the Presentation*
 - During the Presentation*
 - The Final Presentation*

Summary

Key Terms

Review Questions

Critical Thinking Exercises

- Individual Exercises*
- Group Exercises*

References

INTRODUCTION

For the foreseeable future, people will build or purchase software application solutions for use in organizations, often with great effort and considerable difficulty. **Project management** offers the team tools for effective action in four areas: (1) decide which tasks are required to complete the project, when to perform each task, and what the role of each person is in the project; (2) convert plans into action; (3) monitor progress and take action as required to deliver the solution on time and on budget; and (4) use oral, visual, and written media to inform clients and managers about progress and results. The first area addresses *planning*, the second *control*, the third, *execution*, and the fourth *communication*. In addition to such direct activities as project definition, proposed system specification, system design, and system delivery, an effective analyst participates in a number of project management activities. While this chapter specifically addresses project management, concepts of and tools for project management appear throughout the book just as they should appear throughout an entire project.

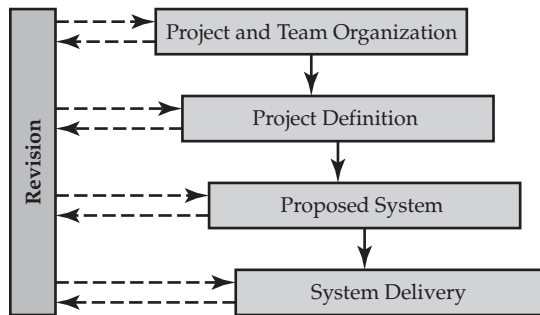
IT projects can involve both strategic and tactical planning and management. Strategic IT planning and management, generally conducted by one or a team of senior executives, addresses how to focus, structure, and fund IT to support the strategic goals of the company. For most field projects, the strategic IT plan exists, often in an implied or assumed form, prior to the initiation of the project. The project team can begin by identifying the strategic alignment of the project as discussed in Chapter 6. From the viewpoint of a client, a successful project is one that contributes to the strategic values of the organization.

This chapter explores the activities and functions of project management within an existing strategic framework. The chapter covers planning, executing, monitoring, correcting, and communicating actions with the goal of realizing “a successful project.” The first part of the chapter discusses project planning including the SDLC approach, the spiral approach, and rapid development. The second and third parts of the chapter look at project execution, control, and communication. The techniques presented in this chapter apply to many activities in addition to information systems analysis and design projects. For an additional discussion of project management, see Forsberg, et al. (2000) or Schwalbe (2000).

PROJECT PLANNING

Project planning represents a fundamental project management activity. Most teams recognize the necessity of planning at the beginning of a project. However, with good project management, planning continues every day throughout the life of the project. Projects face many kinds of uncertainty—activities take longer than estimated, clients change requirements, team members do not possess the needed skills, some parts of planned solution do not work as expected, tools and equipment cause problems, vendors fail to deliver as

FIGURE 3.1
Systems
Development
Life Cycle



promised, users encounter unanticipated difficulties, and many other unexpected events occur. Each time an unexpected event occurs, the team must change the plan. As a result, planning and replanning take place every day throughout the project.

Using the SDLC for Planning

Chapter 1 presented the version of the SDLC that forms the framework for this book using a graphical representation that appears again in Figure 3.1. The traditional **SDLC (systems development life cycle)** was developed in the 1970s as a methodology to plan, manage, and document the process of creating a computer system. An SDLC plan typically consists of a sequence of formal steps that begins with organizing the team and project and continues to the delivery and support of a new system. SDLC-based planning and management brought order and improved results to the large, time-consuming development projects for mainframe applications.

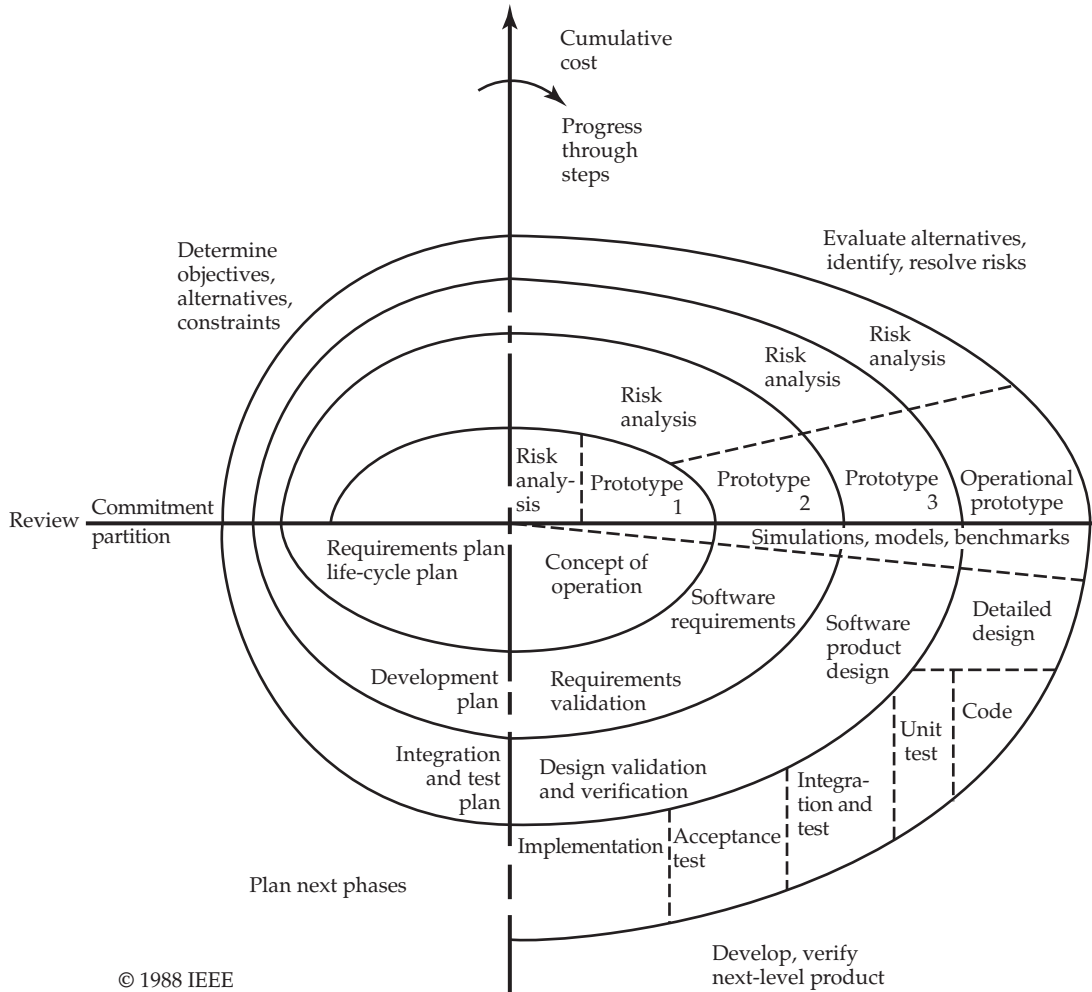
The SDLC developed during the time of large mainframe software development projects when many major computer projects focused on the automation of an existing process to increase efficiency and reduce cost. These projects were plagued by cost overruns, missed deadlines, unmet requirements, and systems that were difficult to maintain and modify. Information system managers believed that many of these problems were caused by the inability of clients and builders to communicate requirements, the inability of project managers to use large teams of analysts effectively, and the difficulty of understanding complex program control logic.

IT managers and clients wanted a planning approach that

- Facilitated communication between clients and builders.
- Provided a basis for evaluation of features and costs throughout the development process.
- Allowed multiple teams to work on a single project.
- Provided good documentation to support operations and maintenance.

A highly structured plan built on the SDLC provided this approach and led to improvement in both the effectiveness and the efficiency of the development process.

FIGURE 3.2 The Spiral Model for Project Management (From Boehm, 1988, used by permission of the IEEE)



The Spiral Model for Project Planning

The **spiral model** for project planning extends the concepts of the SDLC model to recognize that project activities often follow a cyclical path. For example, the team defines requirements at the beginning of the development process during project definition, redefines requirements during the development of specifications for the proposed system, further refines requirements during logical design, and reviews requirements again during physical design. The spiral model, which is shown in Figure 3.2, explicitly presents the cyclical pattern.

The spiral model views projects as cycling through the following four stages:

1. Determine objectives, alternatives, and constraints.
2. Evaluate alternatives, identify, resolve risks.

TABLE 3.1
Comparison
of Project
Planning
Models

Spiral Model Cycle	SDLC
1. Initial Analysis	Project Definition
2. Conceptual Design	Proposed System
3. Logical Design	Proposed System/System Delivery
4. Physical Design	System Delivery

3. Develop, verify next-level product.
4. Plan next phases.

After the evaluation in step 2 of each cycle, the project reaches the commitment partition, shown as a horizontal line in Figure 3.2. If the evaluation team members can justify continuing the project, then the organization “commits” to proceeding with the project. The model shows the option of using prototypes as part of the evaluation stages and using simulations, models, and benchmarks as part of reaching the next level of the product’s development. As shown on the diagram, the spiral model ends with detailed design, coding, testing, and implementation.

The spiral model’s first cycle might be called Initial Analysis, the second, Conceptual Design, the third, Logical Design, and the fourth, Physical Design. Table 3.1 shows the approximate correspondence between the spiral model and the SDLC model used in this book. The spiral model adds levels of detail and structure that may help the team to plan and manage large projects.

Flexible Project Planning

Today, most companies do not follow the SDLC or spiral approach rigorously, if at all. Are companies careless, ignorant, or indifferent to “best practices”? As one might expect in a field where everything keeps changing, the application systems development environment has changed. While highly structured SDLC models worked, they often required a great deal of time and resources. Analysts tried a number of methods to reduce the time and cost of implementing such highly structured plans. **CASE (computer-aided software engineering)** tools were introduced to facilitate and structure analysis and design and to standardize documentation. Automatic table and code generators were added to CASE products to reduce the effort of coding a system after it was designed. Fourth-generation languages (4GLs) were developed to simplify the coding process using a format closer to the language of the original organizational problem. Comprehensive graphical user interface (GUI) development environments attempt to translate the design specifications directly into a product or working system.

In the 1990s, the IT environment underwent further change. The large, complex transaction processing systems that the SDLC and spiral models were designed to build were in place and part of a stable legacy infrastructure. Relational database engines, package application software, and 4GLs were in place to support more flexible and evolutionary trial and error development approaches. New systems to enhance analysis and decision making were designed with a

limited set of functions in smaller projects and with client/server architectures that facilitated decentralized development. The explosion of Internet and intranet applications accelerated these trends. IT groups learned through experience that the traditional plans built on a highly structured planning approach did not fit a number of the new software development projects. The rapid, evolutionary development of smaller projects highlighted the need for more flexible plans.

The switch from mainframe to client/server and Web-based systems also came with a change of development environments and focus on time to market for new applications. Development teams in this new environment were typically young, used to rapidly expanding hardware capability, and raised in a time-driven development world. Many of these teams tried, often unsuccessfully, to code systems without much of any plan. Modern systems require a compromise plan that retains some of the structure of the SDLC/spiral models with the flexibility to produce results rapidly at an affordable cost.

The new **rapid development (RD) planning model** selects parts from and assigns priorities to the SDLC/spiral tasks. As noted in Chapter 1, rapid development represents not a set of methods or tools, but a highly flexible concept: “Do only what is necessary to deliver an application that (1) meets the clients perceived needs, (2) in as short a time as possible, and (3) at the lowest possible cost.” Analysts achieve these goals by incorporating one or more of the following concepts into the project plan:

- Include only tasks or activities that are essential; for example, if requirements are known and the analysts are already somewhat familiar with the current operation, then do not include a detailed study of the current operation and requirements determination tasks in the plan.
- Use programming tools and/or languages that allow fast development and easy changes; for example, build into the plan the use of such tools as DBMSs, SQL, upper- and lowercase tools including database definition and generation tools, Web application design tools, report writers and screen generators, visual languages, objects, and others.
- Work with planning mechanisms that facilitate communication and rapid feedback between analysts and clients; for example, the use of prototypes or client-analyst joint development teams.

The “rapid” piece of rapid development holds special importance. Clients want new applications tomorrow, not in two years. In a global competitive environment, companies cannot afford to wait several years for mission-critical applications; they may go out of business if the new application is not available. Speed of development also can improve the chance of success and reduce cost. With a long project, the problems associated with “scope creep” increase greatly. **Scope creep**, or the addition of features to a system after the initial requirements are set, happens when clients change their minds about requirements or reorganization brings a new set of clients with new requirements for the application. Long development times encourage both clients and developers to add features and refinements that may add little, if any, value. Most changes add cost and can reduce the probability of success.

During the course of a long project, the analysts on the team also may change. Each new analyst starts over, learning what the issues are and what other analysts have done, a costly and error-prone process. When analysts leave during the later stages of a project, many organizations just ask the remaining people to work harder and longer rather than incurring the problems of adding new personnel. The problems and costs associated with changing clients, requirements, and developers are minimized by plans with short development times.

Every aspect of development suggests that keeping the development time as short as possible will reduce cost and increase the probability of success. RD plans work well primarily for this reason. RD plans generally lead to a form of “satisficing,” that is, finding a reasonable or satisfactory approach that will shorten development times, reduce costs, and/or improve the probability of success compared to the full SDLC or other approaches. Unfortunately, RD comes without any guarantees; used incorrectly or inappropriately, it can lead to such problems as higher costs, poor performance, disasters, and more. As always, there are no good substitutes for smart, experienced analysts and clients.

To create an RD plan, the analyst selects only those tasks and activities that are essential to developing the specific application and includes any process, tool, or method that facilitates delivering the application within the RD concept. For a large utility billing system, the RD plan probably will look very similar to the classic SDLC or spiral plan. For a Web site to make parts of an existing database available to employees, the RD plan may consist of one task, “code it.”

Planning Mechanisms

As previously noted, the SDLC provides the starting point for most rapid development plans. The analyst, often in concert with the client, decides which parts of the SDLC are relevant and how to achieve each part. The relationship between the clients for a project and the team that undertakes the effort forms a critical relationship for all application software development and especially for rapid development plans. Because RD plans require some critical decisions, the analyst and client normally make these decisions, that is, determine how to fit the application under study into the RD plan, using a cooperative process based on their prior experience. Teams can choose from a number of mechanisms that facilitate preparation of a rapid development plan.

Client/Team Contract Plans

Under the **client/team contract approach** to RD plans, the client and team, or perhaps a manager or chief information officer (CIO), jointly agree on a broad plan, including the deliverables, major tasks, and schedule for the project. The contract approach offers two advantages. First, because the client is a participant in building the plan, the client is more likely to support the effort and to understand when problems occur and when modifications are required. Second, the client brings additional perspective and experience to the plan. The client often knows how much risk he or she is willing to accept to shorten the development time and/or lower the cost. If a team decides to eliminate one or more major tasks or to follow such an alternative as prototyping-based planning, these decisions should appear clearly in the plan contract.

In practice, most projects and projects teams create either a verbal or written plan contract with their clients. Although verbal understandings may work reasonably well for a small project, the project team can prevent much anger and disappointment by entering into explicit, written contracts with clients. These written plan contracts or agreements often are known as a statement of work (SOW). Guidelines for creating a statement of work are discussed in more detail in the next major section of this chapter.

Planning with Joint Teams

Especially in larger projects, the clients and analysts spend a great deal of time communicating with each other and trying to reach a common understanding about the plan. In 1977, Chuck Morris of IBM conceived ways of getting users together with IS personnel to work out plans to install distributed systems. These ideas evolved into IBM's Joint Application Design (JAD) approach (Wood and Silver [1989], pp. 3–4). The **joint team approach** to RD planning involves forming a team composed of analysts, clients, users, and perhaps managers to work together intensively from start to finish on the project. This approach may work better than the contract approach with relatively inexperienced users or clients. In the contract approach, the plan decisions are made at the beginning of the project, possibly but not always subject to change and modification as events unfold.

Because all the parties are present and actively involved, the joint team approach to RD planning allows the team to make decisions as the project progresses. Many of the decisions and alternatives become clearer as the team completes the parts of the project. In addition to everyone learning from experience, the clients have time to learn from the analysts and vice versa. Having some client members on the team also helps to ensure that design trade-offs reflect both a technical and a client perspective. Clearly, the joint team approach eliminates some of the delays and errors associated with communication between separately located clients and analysts.

Joint effort is most valuable in establishing requirements for the project. If all of the stakeholders participate in a joint design session, they can resolve many of the requirements issues before the IT system is designed. Joint client analyst design sessions are particularly useful when one or more of the following conditions exist:

- Clients are uncertain about what they want.
- Several key clients bring different and/or conflicting requirements.
- Clients have unrealistic expectations.
- The analysts are not familiar with the client's organization.

A possible negative issue with joint teams is the potential cost of taking the team members, particularly the non-IT ones, away from other duties.

Prototype-Based Plans

Prototyping can offer a powerful rapid development tool. The spiral model of system development explicitly includes prototyping. Prototype-based plans rest on a simple concept. In place of or in addition to detailed project definition and

proposed system analysis, the team builds an initial physical version of the application and uses the prototype to answer many of the project definition, proposed system, and system delivery questions. Before starting to construct the prototype, the team must answer some of the project definition, proposed system, and system delivery questions. But in place of a time- and effort-consuming full analysis, the team may use prior experience, read the trade literature, talk with clients, learn from other projects, and/or look out the window and make “reasonable” assumptions.

Prototyping often can clarify many of the issues in the project definition and proposed system phases better than the traditional approaches. After looking at a project plan and standard documentation, the average client and some analysts have only a limited understanding of the implications of the project definition, proposed system, and system delivery issues. Even the most abstraction-challenged clients generally understand at least some of the project definition, proposed system, and system design questions and issues after working with a prototype.

Prototyping works best when a well-designed and understood data structure for the project already exists. If the team does not know the data structure, team members may need to conduct a structured data requirements analysis, for example, building the conceptual data model, as an initial plan task prior to prototyping. Minor changes or refinements in data structure that appear when the prototype is used are simple to incorporate, especially in systems that use a relational database.

Prototypes clearly face limitations. Prototyping may cost more and take longer than expected. A poorly designed prototype can give the client a negative view of the project and can even result in cancellation of the project. The data structure determination may slow the initial iterations to times that the client finds unacceptable. Prototyping can generate the logic and code for a process- or event-driven system, but provides little insight about such physical infrastructure-dependent issues as response time and interoperability. Resolving the issues that the prototype does not address requires placing additional tasks in the plan.

Throwaway prototypes are designed with the intent of throwing them away after using them to determine requirements and/or demonstrate the feel and function of the system. Throwaway prototypes can answer many of the questions for the project definition and proposed system stages of the plan. For this option, the design team produces a sequence of input and output views and the associated logic to demonstrate the way a final system might work. Throwaway prototypes focus on a quick, cheap path to demonstrate feel and function; no attempt is made to provide a program code base for the final system. From a user’s point of view, the I/O interfaces and functions are the system. A throwaway prototype allows a user or client to experiment with the look, feel, and features of the new system. The design team rapidly can modify the prototype to reflect the user or client desires.

Once the client approves the throwaway, the prototype’s only role is to provide the proposed system requirements. The final system may use completely different database engines, programming languages, and infrastructures. In short, the throwaway prototype allows the team to create a “quick and dirty”

proof of concept version of the system without concern for a robust architecture and efficient operation. Once the specifications are determined to the client's satisfaction, the prototype is thrown away and the team proceeds with a system delivery plan, that is, a plan to design or purchase and implement the robust, efficient final system.

Evolutionary prototypes are designed with the intent that the prototype system will evolve directly into the final system. Evolutionary prototypes allow the team to (1) generate answers to project definition and proposed system requirements questions and (2) produce the actual data schema and program code for the system delivery stage of the plan. Evolutionary prototypes use the physical infrastructure, such as database engine, programming languages, and others, selected for the final system. These prototypes may lack some of the complexities required in the final system, but they provide the base on which to add the additional features. Such tools as database engines and 4GLs provide adequate flexibility and efficiency to allow evolutionary prototypes to evolve into robust, full-featured, efficient final systems. For many small field projects, the prototype is the final system.

Outsourcing Plans

Increasingly, organizations acquire part or even most of their IT capability through some form of IT **outsourcing**—purchasing an IT application or service from a vendor. Outsourcing represents another approach to rapid development because it provides a way to reduce time and cost for the solution process. Some organizations choose to focus attention on a few core competencies and then to outsource the noncore activities to other organizations. For example, an oil company might identify its core competencies as (1) exploration—finding new reserves and (2) production—producing oil and gas from existing reserves. The company might perform internally the information system activities related to exploration and production and then outsource such noncore competency areas as human resources and financial systems. Chapter 10 covers a number of issues related to outsourcing.

A decision to outsource changes the tasks and activities in the system delivery phases of the project plan. With an outsourcing option, the system delivery tasks associated with building a solution, such as specify detailed logic, code the programs, and test the code, are replaced with such tasks as to identify vendors and evaluate products or services. A common way to outsource a project is to use a request for proposal (RFP), a document that sets forth for potential vendors the desired system features and constraints and the evaluation process. To use the RFP process, the team sets up the standard tasks in the project schedule for the project definition and proposed system phases. At the system delivery stage, the team defines and executes tasks to prepare the RFP, evaluate the responses, and make a recommendation using the guidelines in Chapter 10. To respond to an RFP, the vendor may perform major parts of the system delivery process, including detail design, coding, testing, and other functions. The vendor selected for the project also may perform or help with implementation, training, and maintenance as part of the purchase contract.

Purchasing a **packaged application** represents a common form of IT outsourcing. With a packaged application, the team and client determine the desired features for the system, but the package designers select the specific features included in their package, design the system, and complete the coding and alpha/beta testing of the product. The tasks for purchasing a packaged solution change from selecting and implementing features for a final design to determining how well the features in each package meet the client's specifications and needs. Because both the designers and other users have already reviewed packages, packages usually offer a more complete and bug-free product than an application that has been built in-house. If the team can find a product that provides the features the clients wants at an acceptable cost, packaged solutions can represent an effective rapid development option.

Sometimes a board of directors or a senior executive mandates the installation of a package, for example, the SAP Enterprise Resource Planning (ERP) system. In these cases, many of the proposed system and system delivery tasks reside in the package; the team focuses on tasks for selecting options within the package, installation, testing, training, and maintenance. Rapid development is a relative concept. While buying and installing an ERP package can take years and millions of dollars, building an ERP system from scratch might take several times as long and use more resources.

Organizations also can implement rapid development by *outsourcing IT functions* to an external organization that (1) can do some or all of the development or (2) can perform some or all of the functions for an area of the organization. Hiring contract workers to execute part of the project changes few, if any, of the crucial development tasks. Experienced contract workers, especially ones with scarce skills, can speed up the development process but may add complexity to project management. Hiring a firm to complete all or part of the project may change the team's major tasks and activities from development to project management.

In common with most rapid development approaches, outsourcing has both potential benefits and drawbacks. Possible benefits include acquiring skills that do not exist within the organization, adding external people on a temporary basis to handle workload peaks, and freeing up internal IT resources to work on core issues. The biggest drawbacks usually involve costs and loss of control.

Generating the Plan

Project plans define the work, identify milestones, and help the team recognize deviations of actual progress from planned or estimated progress. Many team projects run on a tight time schedule with substantial resource limitations. In most cases, the time available to organize a team of people with little common history and then to deliver a useful product is limited. Furthermore, team members have other conflicting duties and schedules to coordinate. The team must organize, pick an appropriate approach, do the work, and deliver a product. The careful design and use of a plan and project schedule can greatly increase the team's performance. Developing a project plan as early as possible also will increase the likelihood of success.

A project plan requires several steps: (1) identify the appropriate activities—task decomposition; (2) estimate person hours and sequence constraints for each activity; and (3) build or generate the schedule in a format managers and team members can understand. While all plans follow this basic structure, the rapid development philosophy means that each system plan will evolve differently as a function of the requirements for each specific project. Plans tend to follow similar paths at the beginning, but the detailed plans will diverge in different directions for different projects over time. The choice of “build versus purchase” represents a major point of divergence, but many other factors cause plans to diverge.

Many times, a team will believe, often with some justification, that the time needed to do the project work correctly or adequately exceeds the time available. To achieve production of the agreed-upon deliverables on time and within budget, every team needs to pay strict attention to plans and to constantly strive to eliminate unnecessary work. Useful planning principles for every project include the following:

- Plans should evolve over time. At the beginning, many of the tasks and most of the problems that will arise during the project are unknown. As the project goes forward and the tasks and problems become clearer, the team can revise the project plan to deal with the unfolding situation in concert with the team manager and clients.
- Good plans reflect actual progress and status. Much of project management involves tracking progress against the plan and revising the plan as needed to adjust for actual progress.
- Teams should revise the plan at least weekly to reflect new or changed tasks and to adjust for actual progress.

Selecting the Activities or Tasks

To identify the tasks associated with a project, the team can begin by identifying the questions that the team must answer to reach an acceptable solution. Each question then defines a task or set of tasks required to produce an answer. Carrying out each task will require the assignment of team members and the expenditure of effort over some period—the start and end times for the task. Each of the tasks with people and the times assigned then becomes an entry in the project schedule.

The SDLC model can help the team to define the questions that the team must answer to obtain a solution to the client’s problem. Although the precise specification of questions, tasks, and deliverables varies from organization to organization, most organizations follow an SDLC type model characterized by major stages and a set of formal management reviews. The SDLC model for this text uses the major stages shown in Figure 3.1: Project and Team Organization, Project Definition, Proposed System, and System Delivery. Each SDLC stage focuses on the answer to a family of questions and involves a set of activities and deliverables. At the end of certain stages, the team may make a formal presentation to a client who may decide to continue, suspend, or cancel the project based on progress and current organizational needs.

As previously noted, the answers to some SDLC questions are predefined. For example, if the client has selected a specific solution, the team may choose to spend little time generating and evaluating other alternatives. Or the client already may have a clear definition for the project and the detailed requirements for the proposed system. A good systems development process performs the activities needed to answer the important unanswered questions that lead to a good solution rather than performs a fixed set of activities. Thus, the project plan evolves from a statement of the questions that need to be resolved at each stage for a particular project and a proposed activity to answer each question.

Typical questions that may require answers and possible corresponding tasks in the project schedule follow.

Project and Team Organization

1. How will the team function? Create the team contract.
2. What skills and resources do the members bring? Develop the skills matrix.
3. What deliverables will the team produce for the client and the manager? Prepare the project schedule.
4. Who on the team will do each task and when? Refine the project schedule.
5. How will the team keep the manager informed? Send the weekly report and set up meetings as needed.

Project Definition

6. What is the problem to solve? Meet with the client and write the project statement.
7. What is the current situation in the organization as it relates to the project? Meet with the client and develop the project definition report.
8. What can the client expect? Prepare the statement of work.
9. Are the team and client in agreement? Conduct the project definition presentation.

Proposed System

10. What are the conceptual-level data, process, and infrastructure requirements for the proposed system? Prepare the proposed system report.
11. What alternatives exist and which alternative is the recommended one? Refine the alternatives, evaluation, and recommendation materials.

System Delivery

12. What are the design specifications for the recommended alternative? Generate the design specifications and/or the RFP.
13. How will the team demonstrate the proof of concept for the system? Find the appropriate prototype, package program, or other model.
14. How will the recommended alternative be implemented and tested? Prepare the testing and implementation plans for the client.
15. How will the team communicate the final results to the client? Prepare the final report and conduct the final presentation.

As time goes on and more information becomes available, the team can expand or modify these questions and the associated tasks. One approach is to generate the detailed tasks for each new SDLC stage or deliverable as a final part of the deliverable for the preceding stage. For example, the team can refine and schedule the detailed proposed system stage tasks as a final part of the project definition work. An astute team reviews or works jointly with the client and manager as appropriate on all significant plan modifications.

Within the schedule, client meetings deserve extra attention. The team may encounter significant difficulties in getting adequate access to and feedback from clients. The team should plan as many client meetings as needed and try to schedule them at the beginning of the project. Getting onto a client's schedule at the last minute is difficult. If meetings are scheduled well ahead, clients can contact the team coordinator if conflicts arise, and hopefully will feel some obligation to reschedule the meeting. Whenever possible, teams should get the work done before scheduled meetings with the client or manager and not try to arrange a last-minute meeting when the work is done.

In real-world IT practice, the steps in the schedule may differ greatly from the SDLC. A real-world team pursues a single goal: the successful, fast, and cost-effective development of the product that the client wants. The team reviews the problem and selects only the steps that are needed to meet the RD goal. Some companies use formal prototyping methodologies, trial solutions, and little or no other documentation to accomplish many of the functions in the SDLC. Consultants and company IT groups may develop special standards and methodologies to shorten the life cycle. The company may provide guidelines, for example, informal plans for small projects and SDLC-like plans for very large, complex projects. In non-IT companies, multiple levels of management may review and attempt to influence a plan especially for large projects. In IT companies that specialize in software development or in large IS shops, project leaders and/or team members may possess the trust, skills, and experience to create a plan with few if any guidelines and little supervision.

In a student field project course, the activities in the plan must address at least two objectives: (1) allow students to demonstrate the joint application of theory and principles from business, programming, database, infrastructure, and system analysis courses; and (2) deliver a useful product to a client. The education goal may involve activities that are of little interest to the client. As a result of the dual goal, the team probably will include more steps in the plan than a company IT team would use for any but the largest projects. The cost of education can be high in many ways; hopefully, the benefits are commensurate.

Task Times and Sequence

Once the team determines specific **tasks**, the next step is to define a **time** required for each task and the sequence constraints. The team can put the tasks into sequence by identifying the other tasks that must be completed before the current task can start. **Sequence constraints** should reflect only requirements and not tradition. For example, building a proof of concept model appears in the system delivery phase of the SDLC, the last stage. However, often the team can

and should start the design or selection of the proof of concept model near the beginning of the project because it is a long lead-time item. On the other hand, prior to visiting the client, the team may find starting the project definition DFD offers little value.

Each activity has two sets of relevant times: (1) the start and end (due date) calendar times for the activity; and (2) the number of person hours of work required to complete the activity. The two are related but often not directly. If an activity assigned to one person for an hour of effort due in a week actually takes two or three hours, the effect on the due date may be negligible as long as the person starts the activity before the last minute. The main effect of underestimation in this example is an increase in cost. If the activity due in a week is estimated to take 40 hours of effort and actually takes 120, then both cost and due date may be affected.

Estimating time requirements represents a complex activity. PERT/CPM methodology includes statistical procedures to estimate time requirements as well as a process for identifying critical time deadlines that impact final delivery. IS developers have used such parametric estimation approaches as function points, lines of code, and COCOMO to predict the time required by a project with mixed results (see Schwalbe [2000], pp. 151–153 for a discussion of estimation procedures). All of these approaches share some common properties: They require experienced planners and often a database of past estimates compared to actual times. Project estimation remains an art that relies heavily on experience with similar projects.

With less experienced analysts, simple projects, and no database, a reasonable approach goes as follows. Use the experience of team members and any other available sources of experience, such as managers or colleagues, to make estimates. When making estimates, allow for the learning curve or ramp-up time effect. Coding in Visual Basic or using ASP for Web site design may require one or several team members to spend extensive time learning a new technology. Recognize that the times may be incorrect; they are most often underestimated. To compensate for the tendency to underestimate, allow more time. Start early to keep time overruns from interfering with due dates, double or otherwise increase your first estimate, and be prepared to replan as needed.

When actual person hours to complete a task exceed the estimate and will extend the completion of the task beyond a due date or critical path time, the team needs to take immediate project management action. Possible actions include adding additional resources or people, working longer hours, looking for a way to reduce the size of the task, revising the schedule, and/or asking the manager for an extension. The key point is to address the time misestimates that interfere with due dates as they arise. Hoping and praying that the problems will go away or that no one will notice works poorly in the development environment.

Constructing the Schedule

Once the tasks, sequences, and time estimates are generated, the team can construct the project schedule. A complex system may require the use of a project management tool to build the schedule and identify the critical path. For

moderate-size projects, an activity table or a Gantt chart offers helpful graphical model representations for the schedule (Schwalbe [2000], pp. 118–122, presents a description of the use of a Gantt chart).

Suggestions for the assignment of start and finish times to activities include the following:

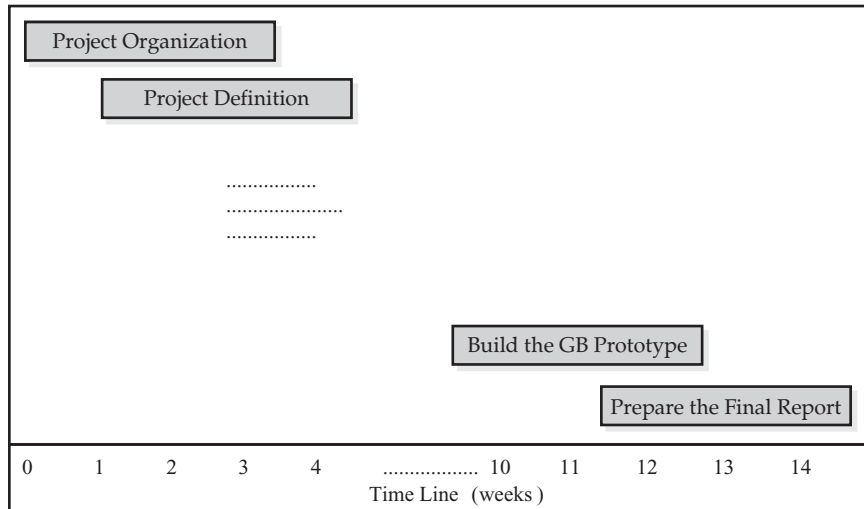
- Start by entering any predetermined times, particularly the starting time for the project and the date that the final deliverable is due. All the work must fit within these two dates. Many projects will have other interim due dates. The typical field project course, for example, has a number of due dates.
- Schedule as much work as possible as early as possible in the project. This approach leaves time for the inevitable slippage. A few activities may take less time than planned, but many tend to take longer than estimated. And the team, the manager, and the client will identify new required activities as the work proceeds.
- Note that some activities have few sequence constraints. For example, the coders can learn a programming language for the prototype as soon as it is chosen; they do not need to wait until the team completes the pseudocode and data schema. Team members can search for possible package solutions after the first meeting with the client; they should not wait (and do not need to wait) until the proposed system report is finished. Again, start work as early as possible.
- If possible, build in some slack or unused time before each due date in case the work takes longer than planned. Most teams find that they often need the slack to get the work done.

When assigning people to tasks in the schedule, the team should consider workload balance and concurrent activity. If the schedule calls for 180 hours of work in one week for two people, then the two either will have to give up eating and sleeping, or they will miss the deadline. Goals include not overloading any one person and assigning roughly equal loads to each team member. Assigning one person to start a task and another to finish it often causes coordination and blame-shifting problems. Assigning the same person(s) to a task from beginning to end generally works better.

The graphical format used for the plan can vary depending on the preferences of the team, manager, and client. Often the client wishes to receive only summary information on progress and a list of delivery dates for the key deliverables. Thus, the manager and the team often determine the format of the detailed schedule. Many IS organizations have a standard format for detailed project plans, for example, “Generate the plan using MS Project with options *a*, *b*, and *d*.” In less structured situations, the team and manager may select either a Gantt chart or table format for the plan.

Figure 3.3 shows a sample RD plan as a **Gantt chart schedule** for GB Video. The chart contains only a few tasks; a typical field project plan will contain more and perhaps different tasks. The Gantt chart allows easy visualization of the start and end times and a sequence for a limited set of events, but it becomes

FIGURE 3.3
Project
Schedule in a
Gantt Chart
Format



confusing with a large number of activities and extensive detail. The chart may require a supplementary set of notes to record such data as people assignments, due dates, and estimated person hours. Finally, without a computerized scheduling program, updating a Gantt chart can require a lot of work.

An activity schedule table offers an alternative that is easier to manage than a Gantt chart. The table might show

- The major milestones or stages for the project.
- The key activities associated with each milestone or stage.
- The estimated or planned person-hours for each activity.
- The people assigned to each activity.
- Sequence prerequisites, the other activities that must be completed prior to this activity.
- Key dates for the activity, such as
 - Start date—time the team plans to start the activity.
 - Draft date—time to start preparing a draft version of a deliverable.
 - Due date—time that a deliverable is due to the manager or client.
- Status or condition of the stage or activity, for example, not started, scheduled, started, done, completed, turned in, in draft.

A sample **activity schedule table** is shown in Table 3.2. As before, the example is incomplete and shows only a few of the entries needed for an actual field project. A schedule table, however, can contain a lot of information and the team can change or update the table with ease in any computer word processor. Note, though, that sequence and the interrelationships of start and stop times are more difficult to visualize than they are in a Gantt chart. Again, the team and manager work together to decide the specific schedule format or combination they wish to use.

TABLE 3.2 Project Schedule in Activity Schedule Table Format

Milestone or Activity	Est. Person hours	People Assigned	Sequence PreReqs.	Start Date	Draft Date	Due Date	Status
1 Organization Plan				13 Jan	19 Jan	21 Jan	Started
1.1 Deliverables							
1.1.1 Team Contract	10	All		13 Jan	19 Jan	21 Jan	In draft
1.1.2 Skill Inventory	5	All		13 Jan	17 Jan	21 Jan	Complete
1.2 Meet with Manager	6	All	1.1	20 Jan	20 Jan	21 Jan	Scheduled
1.3 Meet with Client	18	All	1.2	22 Jan	23 Jan	24 Jan	Scheduled
2 Project Definition				24 Jan	31 Jan	3 Feb	Not started
2.1 Review Client Request	6	All	1.3	24 Jan	24 Jan	24 Jan	Not started
2.2 Draft SOW	8	Al/Dick	2.1	25 Jan	31 Jan	3 Feb	Not started
2.3 Presentation	30	All	2.1	24 Jan	31 Jan	3 Feb	Not started
3 Proposed System							Not started
4 System Acquisition							Not started
5 System Delivery							Not started
5.1 Final Report and Presentation				9 Apr	18 Apr	23 Apr	Not started

STATEMENT OF WORK

The **statement of work (SOW)**, as discussed earlier, represents one approach to a rapid development plan. The SOW defines the work that the team will perform for the client in a written statement signed by both the team and the client. It can include a description of the potential solutions that the team will pursue and can identify the resources that are expected from both the client and the analysts. The statement of work can represent the final step of the project definition stage.

A typical statement of work might include the following elements:

- **Description.** A description of the project—a short description of the problem the client asked the team to solve.
- **Work product.** A description of the work that the team will perform to address the client's problem. For example, the work product can survey users, determine the detailed requirements for the proposed system, design a data structure, specify the processes for the system, build and demonstrate a prototype, develop specifications for a package, identify possible packages, evaluate packages, recommend a solution, implement a solution, and more.

The work product section also lists the classes of solutions that the client and team agree to consider. Possible solution classes include to build the system in-house, hire a contractor to build the system, purchase a package, outsource the information system operation or the entire function, and keep the current system, perhaps with modifications. Unless there is a strong reason to the contrary,

the team must evaluate the merits of at least two possible solution classes as part of the work.

This material should contain enough detail to allow the client to make a decision on funding or continuing the project. The descriptions may include specifications for such things as performance levels, evaluation criteria, and documentation standards. In all cases, the work product section should define clearly the project scope. If the project includes delivery of a prototype or a package, describe the prototype or package in enough detail to define its scope.

Often, the initial scope decision is made or implied by the client as a result of specifying the goals for the new system. Look carefully at the scope and refine it in concert with the client. Complexity as measured by the hours spent on a project increases rapidly with size. Within the constraint of including those areas that are important to the client, keep a project small enough to fit the available time and resources. In keeping with RD concepts, drop from the project scope any areas that may be “sort of interesting” but not really important or central to the client’s perceived needs.

As part of the work product description, note the major deliverables that the team will provide to the *client* (not to the manager). Most projects will have only a few client deliverables, for example, a project definition review, a prototype demonstration, and a final presentation and report.

- **Client resources.** Define in the SOW any client resources that you expect to use for the project. Client resources usually include the client funds, existing infrastructure, and people available for the project. Resources also may include permission to interview people, use of client IT people or resources to help you with the project, client-supplied test data, permission to use the company’s name in contacting outside vendors or copies of software or data.
- **Project success criteria.** A careful, explicit definition of the criteria that the client will use to determine the success of the project. The criteria may include completion time, budget, quality, completeness of deliverables, and the impact on organizational performance.
- **Project schedule.** A plan in table or chart format that gives the client the date for every key event that will involve the client—client deliverables and major planned visits—for example, your final presentation. One of the critical problems for projects with short deadlines is assuring access to busy clients. Arranging appointments at the beginning helps the client to reserve the time.
- **Signatures.** Normally, the lead client and one or more team members sign and date the SOW. While not a legal contract, the SOW is a professional agreement in which all parties pledge to use their best efforts to meet the terms.

In small projects, the SOW probably will fit on a couple of pages. In a large project, the SOW may require many pages. A sample statement of work appears in Figure 3.4. A well-done SOW for an actual field project probably will contain more detail than this example.

FIGURE 3.4 Statement of Work for GB Video Company

Team 7 Statement of Work for GB Video

Project Description

GB Video asked Team 7 to design a new computer-based system for rental and return of videotapes and DVDs because the current manual system maintains inadequate records and is too slow. The new system should address improved customer service and lower handling costs for each transaction.

Work Product

The team proposes to perform the following work for GB Video:

- Conduct a strategic analysis of GB Video to determine mission, objectives, goals, and performance measures.
- Analyze the current video rental and return situation to (1) identify goals for the proposed system and the constraints that GB Video wishes the team to observe, and (2) understand the current operation, system, and problems. The team will review these materials with GB Video before proceeding.
- Develop data, process, and infrastructure requirements for a new computer-based system. The system scope includes video rental and return and enrolling members.
- Create and evaluate at least two design and procurement alternatives.
- Recommend a viable approach to solving the problem,—such as a recommended design and procurement option for a new computer-based system for rental and return of videos. The team understands that GB Video does not wish to keep the current system unless all other alternatives exceed the budget constraint of \$200,000. With the agreement of GB Video, the team will consider both build and purchase options.
- Demonstrate a proof of concept model—either (a) build, document, and demonstrate a prototype rental/return system or (b) create an RFP to select and demonstrate a purchase option for a video rental/return system.
- Provide a final report to GB Video that sets forth the essentials of all the work carried out by the team and containing sufficient detail to allow GB Video to use the report as a detail design and procurement document for the new system.
- Make a final presentation to GB Video to highlight the key points in the report.

Important Note: The team is not planning to deliver a final operational version of a rental/return system to GB Video. GB Video either can perform or contract for the production or procurement, testing, and implementation of an operational computer-based video rental and return system based on the design and analysis provided by the team.

Project Success Criteria

The team and client agree that the success of the project will be measured against the following criteria:

- The team delivers the final report to Mr. Cosier by no later than 23 April of this year. This time criteria is extremely important and any delay will be considered a serious failure by the team.

- The recommended solution contains all of the features requested by Mr. Cosier and is within the constraints. If for some reason the team cannot meet this criteria, the team will immediately inform Mr. Cosier and include a complete and detailed analysis of the reasons for the problem.
- The quality and completeness of the analysis and evaluation leading to a recommended solution meet high professional standards.

Initial Project Schedule

The tentative dates for project activities include:

January 24	Collect initial information from GB Video
February 6 at 4:00 p.m.	Project definition presentation
April 15	Demonstrate the proof of concept model
April 23	Deliver the final presentation and report to GB Video

Note: The team expects to schedule other meetings as needed and will arrange the dates with Mr. Smith as far in advance as possible.

Client Resources

Mr. Cosier has agreed to provide an assistant manager, Robert Smith, as the contact point. The team will always call for an appointment and not just drop in. Mr. Cosier also will provide a suitable server, network, and workstations if the decision is to build or buy a package.

Signatures

For Team 7

Al Price

Date _____

For GB Video

Roberto Cosier, President

Date _____

The context for the SOW shown in Figure 3.4 comes from a hypothetical case called the GB Video Company. The GB Video project materials are used throughout this book for illustrations and examples. In the case, Mr. Cosier, the president, gives an IT development team identified as Team 7 this instruction: “Design a new computer-based system for the rental and return of DVDs and videotapes because the current manual system maintains inadequate records and is too slow. The new system should address improved customer service and lower handling

costs for each transaction.” See Appendix A for the full set of the Team 7 project materials associated with GB Video.

Many organizations use some variation of the SOW approach to facilitate rapid development. SOWs help in establishing the scope of a project, reducing misunderstandings, and eliminating unnecessary activities. The effective use of a SOW requires clients who can make meaningful choices about development and analysts who can and do inform the client on technical issues. For example, the analyst has an obligation to help the client understand the issues about the best language or tools to use and the risks of using a current or proposed infrastructure for the new project without extensive benchmarking and testing.

Even with a SOW, a number of problems remain. Having an SOW will neither keep requirements from changing nor guarantee that clients understand the agreements. The project team will have to revise and review requirements and expectations throughout the contract. The team also will have to recognize when changing circumstances (1) require more time and resources than are available or (2) make the SOW agreements infeasible for some other reasons. In either event, the team must immediately discuss the problem with the manager.

PROJECT EXECUTION AND CONTROL

Project execution and **control**, two closely interrelated functions, start with the process of converting plans into action, that is, put team members to work on the various activities needed to meet the plan. Project execution and control operate in the context of a feedback control system: take initial action, observe results, compare progress to the plan, take action as needed, observe the results of that action, and so on. The team uses the project control activities to identify problems and opportunities. The team can then take new actions to alleviate the problems, capitalize on the opportunities, and use the project control function to monitor the results. The only thing certain in a project is that things *will* change.

To deal with change, the team needs a mechanism to collect information on actual required tasks and progress, and then to compare the actual to the plan. If the actual situation departs from the plan in a manner that can affect completeness, correctness, or due dates, then the team should take corrective action. The appropriate actions may include adding, deleting, or revising tasks; revising schedule times or sequence, changing the assignment of people; asking people to work more effectively and/or longer hours; changing tools and technologies; working with the client and/or manager to revise due dates, deliverables, and/or scope; and, when necessary, canceling the project. Selecting the appropriate actions to take, if any, as a result of departures from plan presents one of the major challenges to project management.

The team can select from among several approaches to project management. The team members jointly may perform the project management functions or the team may assign responsibility for project management to one or more members. Either approach or a combination thereof will work; the critical issue is to make sure that the project management functions occur promptly and effectively.

Good project management starts at the very beginning of a project and continues until the very end. Time is often the scarcest and yet most critical resource for the team. Field projects generally receive sufficient funding (i.e., the efforts and other resources provided by the team members) and adequate organizational support, but they almost never contain enough time. At the beginning, the end of the project seems a long time away. The team easily can lose a lot of time early in a project on confusion, low-priority activities, or other wastes of time only to discover that the team cannot finish the actual product creation by the deadline. A team that uses the first few weeks well and gets off to a good start significantly increases its chances of success.

Project Execution

A plan is a piece of paper or some data in a computer. Even the best plan produces no effects until the symbols in the plan become actions. The first project management task is to make certain that all the team members know what they are supposed to do and when, that is, to convert the plan into work performed by members of the team. The team should make a special effort to eliminate confusion, misunderstanding, or ambiguity at this time. Putting all assignments in writing can reduce arguments about who was supposed to do what.

A poor execution of a plan might take the form of, “Al, since your name appears on the plan for the project definition report, you work on it.” If Al spends an hour thinking about the report, has he performed his assignment for the week? Or, does Al have to collect information from the client, write the text, draw the EDM and DFD, edit the report, and submit the finished copy to the manager, all by himself? The team probably intends something in between; but unless the team-members provide more specific and clearer project execution direction, the results may both astound and disappoint them. At every meeting the team should check to make certain that each member’s view of his or her assignment corresponds to the team’s view.

Controlling Changes in Operations

IT projects are all about change. An organization sets up a project to bring about change—to modify the current operation by the introduction of a proposed system. The resulting changes may impact all of the stakeholders, including owners, managers, customers, clients, users, and IT people. Change always brings the possibility of resistance from one or more of the stakeholders. The following are some thoughts about **resistance to change** and possible mitigating actions.

- People resist change when they are uncertain about and/or fear the consequences. The team deals with this issue by trying to provide good information. For example, a proof of concept model, by demonstrating the feel and function of the system, may alleviate (or reinforce) user and client concerns far more effectively than handing them a written description of the new system.
- People resist changes that offer a perceived negative benefit or negative reward to them. For example, workers may fear the change will reduce their

usefulness or comparative advantage or increase the amount of work they must do with no direct benefit or make their jobs less interesting. The team may deal with part of this problem by recommending to the client such things as (1) training for the affected group, either on how to use the new system or on how to perform in a new job area; (2) changes in the work content of a job; and (3) changes in rewards or other factors that eliminate or, at least mitigate, the perceived negative benefits.

- People resist changes that they believe do not solve the perceived problem. If users or clients appear dubious, the team should recheck their presentations of the system features and benefits and/or check to see that they are addressing the correct problems and that their solution makes things better.

Controlling Changes in Requirements

Changes in system requirements come from at least two sources: the team and the client. Changes requested by the client are a form of change that can produce a major impact on the team. Virtually all IT projects undergo some client-proposed changes in requirements as the project proceeds. Most of these changes cause additional work for the team and can jeopardize the prospects for success.

A key project management task for the team is to control the amount and timing of client changes in requirements without antagonizing the client. Clients prefer great freedom to make changes in requirements whenever they want throughout the project. The team wants to freeze the requirements as early as possible. The best compromise comes from sharing a thoughtful cost-benefit analysis with the client. If the team can demonstrate the cost of a change to the client, whether it is a delay in completion time, a risk that the project will not succeed, or other cost, then the client can make a rational decision on whether or not to proceed with the change. The team carries a heavy obligation to provide a fair analysis; a team that routinely announces that every proposed change is prohibitively expensive soon loses credibility.

During the initial project definition stage, changes in requirements pose few if any problems as long as the total project scope remains within the limits of available time and resources. In fact, the team wants to encourage the client to consider requirements changes at this time. The project definition presentation represents a formal opportunity for the client to review scope and request changes. Throughout the project, the team probably can and will incorporate some changes to the design, for example, the proof of concept demonstration to the client is intended to stimulate client thought about the completeness and appropriateness of the design.

The later in the project that a new client issue surfaces, the more the team needs to exercise care to prevent the scope from expanding to a size or into a functional area or technology that the team cannot handle with the resources available. Once in the system delivery stage, the team must carefully evaluate the impact of all changes before agreeing to them. Sometimes a change will require only minor work on a program, database, or RFP; other times the change will require the team to almost start over or to increase greatly their time commitment. The team

should contact the team manager immediately if the client insists on an unworkable or unreasonable expansion or change of the scope.

Changes proposed by team members also can cause problems. As the project proceeds, members of the team will discover possible ways to improve the solution. However, the team always should ask what impact a change will cause on the quality and due dates of their deliverables. Some changes improve the work product and result in little or no impact on effort and due dates. However, other changes, even though they may significantly improve the deliverables, may also cause havoc with workload and due dates.

Because of the need for consistency, introducing a change into one part of a project may require changes to many other parts. Each change increases the probability of errors in the final deliverables. A satisfactory prototype that works and is completed on time may serve the team and the client better than a “great” prototype that does not work or is finished past the due date. Sometimes most suggestions for change are unwise. When a new junior programmer joins an actual project team, the programmer is expected to give first priority to writing programs that implement the current requirements, not to trying to change the requirements. A policy of staying with original decisions as long as they produce a satisfactory outcome has much merit.

Monitoring Progress against the Plan

Managing change dominates IS project management. Each day, the team discovers tasks to include in or remove from the RD plan. Tasks take more or less time than expected, some people do not do what they agreed to do, and other problems occur. The key to success in project management lies in obtaining and using information on progress. Some teams try exception reporting, reports that let the team know when and only when something changes. However, the people most likely to fall behind are the people least likely to report. Spending time on a careful review of progress at each team meeting probably represents the most practical approach.

The most reliable status report is one that says an activity is either complete or not started. Estimates of percent completion, for example, “I am 80 percent done with the prototype,” are notoriously unreliable. The last 20 percent (whatever that means) may take more time than the first 80 percent did. Breaking tasks into subtasks often provides better information. For example, instead of saying that a prototype is 80 percent complete, one could say, “I finished creating and populating the prototype database and am ready to start creating reports.”

Once the team understands the current status of a project, the team can **monitor progress** by comparing the actual progress to the plan. The schedule chart or table provides a good tool for comparing actual and plan. By redoing the schedule to reflect current status, the team can see what appears to be happening to completion times for deliverables. If the completion time gets close to or exceeds the due date, then the team needs to take corrective action. Many organizations require project teams to submit updated plans on a regular basis as part of the project management control system. For large projects, a computer-based

project management tool, such as MS Project, can facilitate this kind of updating and reporting.

Taking Corrective Action

Corrective action represents a most difficult aspect of project management. When a team is in serious trouble, it will go to see the manager. Team members explain that they fell behind a month ago and now have no hope of finishing on time. When asked what actions they took to correct the problem, they look surprised and offer no answers. Even when a team recognizes a problem, the team often takes no action until it is too late. At every team meeting, the team should discuss not only the status of the project, but also the actions that might correct any problems.

Possible actions to correct problems that arise when actual progress falls behind planned progress include the following:

- None needed except to change the expected completion date—the activity is not on the critical path and thus has enough slack to prevent any delay in the project.
- Reallocate resources to add person power to a critical area that is falling behind. One or more team members may be working on a noncritical task and can be reassigned to a more critical one.
- Work more effectively. With some thought and analysis, people often can use their time better, that is, spend time on the most important activities, seek help for tasks they do not understand, and so on.
- Work harder. Most people already have more than enough to do, but in emergencies they can put in more effort.
- Find the best person. Sometimes the team members assigned to a task discover they lack the needed skills while another team member working on a different task has the needed skills. Clearly, the team should change the assignments at this point.
- Find a consultant. If the team members face a problem for which they lack the needed skills or knowledge, the team should move rapidly to get outside help from colleagues, the client, the manager, or anyone else who can help. A project is not a closed-book exam; teams are encouraged to consult with others.
- Reduce the scope. Sometimes the team can make the decision by itself, for example, to decide to add fewer bells and whistles to the prototype. Other times, the team will need to discuss the scope change with and receive the agreement of the client and manager.
- Ask the manager for an extension in a due date. Appeals should come with a good reason. Appeals also should come as early as possible and well before the due date. When the team asks for an extension an hour before the due date for a problem that occurred a month ago, the granting of the appeal is unlikely.

When a problem first occurs, a number of the above actions may work. A month later, none of them may work. A key to effective correction action is take action as soon as the problem is detected.

Project Review Points

Although a team can and should take action any time a problem appears, most organizations also identify and conduct in-process reviews at key points during the project. For example, the spiral model contains a commitment line. Each time the project arrives at the line, a **review** occurs to determine the appropriate action: proceed on plan, make changes in the plan, or cancel the project. Each organization may have a set of points at which reviews of the project by the client occur. Possible major review points are listed below.

- *The Project Definition Review.* The team reviews their understanding of the current operation and the goals and constraints for the proposed system with the client. This review often clarifies the team's understanding of the current operation and problems and may result in changing goals, constraints, scope, and deliverables for the proposed system. The review may include the contents for a statement of work.
- *Proposed System Specifications.* After the team has developed specifications for the proposed system, developed alternatives, and selected a recommendation, the team reviews this work with the client before proceeding with a build or procure option. The client may ask the team to change the specifications and recommendation or to pursue multiple options.
If approval is granted at this stage, the team proceeds on the path to build or procure a solution. Once the team has good information either from review of the possible procurement options or from building a prototype, the team is ready for the final design review.
- *Final Design Review.* With a procure option, the team and client make a decision at this point on the specific procurement option. With a build option, the team may have a prototype or initial version of the system plus refined estimates of cost and time to complete the project. If the expected outcomes are not satisfactory, the project either ends or starts a search for new alternatives often with a revised scope. Once the project proceeds beyond this point, major expenses are incurred. Projects should be terminated or revised at this point if the available options do not meet the desired performance specifications or if time and expense forecasts exceed constraints. For many student projects, the final presentation basically is the final design review.
- *Pre-Implementation Review.* By this time, a full operating version of the system exists in-house and has undergone testing. The team and client review the results and make the decision to proceed with implementation or to conduct modifications and/or further testing. Projects rarely are terminated at this point. However, if the new system when implemented has a significant chance of disrupting operations and causing expensive problems, the client and team should seriously consider termination. While losing the investment to date in the system is bad, incurring large additional expenses from a system that does not work satisfactorily is worse.
- *Post-Implementation Review.* Systems seldom perform exactly as expected. Some features may not work in the client's specific situation, users may find ways

to use features not anticipated by the designers, and finally, the environment in which the system operates may change. The post-implementation review determines how the system actually is performing. The review can identify successes and problems, focus corrective action as needed, and provide valuable insight for future projects.

In addition to reviews that involve the client, the team may conduct a number of individual and team reviews. For example, the team members normally will conduct a team review of their work prior to every review with the client. Problems found and corrected by the team cause far less embarrassment and potential punitive actions than problems found by the client during a review.

PROJECT MANAGEMENT TOOLS

Project management involves a number of time-consuming jobs. The team members must identify project tasks, estimate durations, identify dependencies (when some tasks can be performed only after others are complete), assign resources to the tasks, and organize all this information in Gantt charts, PERT charts, activity schedule tables, and other formats. Constructing the initial charts and tables is a daunting assignment; however, for project management to work correctly, the team must update the schedule data frequently (often daily) as new tasks appear and task dependencies, completion times, and resource assignments change. A team easily could spend much or most of the available project time manually updating the schedule with little time left over to solve the client's problem.

Fortunately, computer-based **project management tools** can perform many of the routine tasks. These tools provide convenient formats for input data on tasks, durations, dependencies, and resources; create an initial schedule; update the schedule in response to change data; and generate a variety of analyses and reports. The effectiveness of a project management tool depends heavily on the team's diligence in identifying and reporting changes. If the team neglects to identify and enter change data in a timely and complete fashion, clients and managers viewing the reports will receive an incorrect impression of progress and problems, and team members may spend time and effort on noncritical tasks while critical ones go unattended.

Teams can choose from literally hundreds of project management tools. The tools require an initial purchase and, more important, an investment by users in learning how to use the tool. However, most of the tools are relatively inexpensive and relatively easy to learn. Many of the tools work on a PC, and a number of tools provide networked versions that allow a group of users—the team members, clients, and anyone else whom the team grants viewing and/or updating access, to view the schedule and other project reports from any Web-enabled workstation and, with updating rights, to make changes. Often an organization selects a preferred tool for all the teams in the organization to use. Industry surveys suggest that Microsoft Project (see <http://office.microsoft.com> for more information) is the most widely used tool, but a number of others also are used.

Microsoft Project provides a familiar Windows interface for entering input data. The team enters the list of tasks, sets the task durations, and creates dependencies between tasks. The tool allows the team to specify lead times (e.g., to start task A when task B is 50 percent complete) and lag times (e.g., to allow a two-day gap between the end of task B and the start of task C). The team also can assign resources, including people, to tasks, and each resource can have a unit cost. Microsoft Project will display the schedule in Gantt chart format or in a Network diagram (**PERT chart** format). MS Project will identify the “critical path.” When an activity is on the critical path, any delay in completing the activity may delay the entire project. Clearly, the team and the client want to focus attention on the activities on the critical path. MS Project also can produce a variety of other reports on project activities, costs, and resources. As the project proceeds and things change, MS Project updates the charts and reports when the change data are entered. Microsoft Project is available in both stand-alone PC and network versions.

PROJECT COMMUNICATION

Project communication with team members, managers, and clients forms an essential part of every project. Communication with the client generally occurs on an “as needed basis,” although sometimes clients request regular status reports. Communication with the team manager often occurs on a scheduled or regular basis, perhaps weekly and whenever a significant unexpected event occurs. The team can use a variety of forms of communication: face-to-face, telephone, e-mail, or delivery of written documents. The best form to use depends on both the people involved and the content. The team can ask both the client and manager about which forms of communication are acceptable and preferred. For example, teams can complement written progress reports with periodic meetings with the manager. The purpose of these regular meetings is to give the manager and team time to review work and discuss issues. Normally, the team can deliver routine messages and reports by e-mail or telephone; the delivery of significant, bad news, however, works best with a face-to-face visit.

The following material addresses communication with clients and managers. Chapter 2 covers member-to-member communication within a team. During the course of a project, the team normally will report progress in writing or in electronic form on a periodic basis. As milestones are reached, the team will make oral and written reports of substance to the client. The team concludes a project with a final report and a final presentation for the client. A working copy of the program and related documentation is part of the final report. If the organization uses a standards manual, all communications should follow the guidance in the manual.

Progress Reports

Effective management of a project requires good information on progress or a **progress report**. Typically, managers monitor progress by tracking departures

(exceptions) from plan, critical success measures, or a combination of the two. Exception reports to the manager provide information on only actual or potential departures from the plan, that is, delays, difficulties, or potential problems. In the critical success factor approach, the manager specifies the measures of interest, such as the completion of key activities or deliverables, client satisfaction levels, total team effort, and other measures. The team reports the values of these measures. Most managers and clients prefer some combination of information on both departures and success measures.

Timing and frequency represent additional issues in reporting. One option is event-oriented status reporting, whereby the team reports all significant changes or accomplishments to the team manager when they happen. The team may update the RD plan as events happen and give the manager access. Or the team may send written or electronic notices to the manager. For many projects, event reporting provides more timely and voluminous information than the manager wants. Normally, progress monitoring on a weekly or longer basis can provide all the information a manager can use.

A typical **weekly report** contains: (1) a text summary of significant team accomplishments; (2) any problems and the actions the team is taking to solve them; (3) plans for the next week or next several weeks; (4) a table that *for each person* on the team shows: (a) work performed this week; (b) hours expended on the project this week; and (c) total hours expended on the project to date; and (5) an updated RD plan showing work completed to date and project status (e.g., on schedule, ahead of schedule, or behind schedule). Figure 3.5 shows a sample weekly progress report. A team that takes rapid development seriously will modify the last week's report to create the report for each new week. The team can create the format and tables once and modify them as needed.

The hours reported for each person for the week should fit within reason the text description of the work performed. For example, a report of 40 hours to arrange a visit to the client seems unreasonable; one hour for this task seems more reasonable. If the person actually spent 40 hours on the task, the person should include a note of explanation. Report only the time spent on the project; however, time spent reviewing the materials for project guidance or skills comprise an appropriate charge to the project.

Written Reports

The team may prepare several interim written reports and a final report for the client to record the results of the team's analysis and design activities. The reports help the client to evaluate the team's work and provide a base for any additional or future work including system operation, maintenance, and modification. The contents and sequence of the reports follow the organization's standards, if any standards exist. Many reports start with a description of the problem that the client asked the team to address, then describe the team's accomplishments to date, and end with the work required in the future.

A sample list of contents for a final report appears in Figure 3.6. The materials in the report, except the table of contents and the executive summary, are discussed in detail in subsequent chapters. Interim reports will follow a similar format but

FIGURE 3.5 Progress Report for the Week of January 24

Memo

To: Team Manager
From: Al Price for Team 7
Subject: Progress for the Week ending 24-Jan
Date: 1-27

Significant Team Accomplishments

This week, the team turned in the Team Contract, held its first meeting with the client, GB Video, and started work on project definition and the SOW. During the meeting, the team discussed scope, the strategic environment, current operations, problems with the current system, and goals for the proposed system. We scheduled 6-Feb at 4:00 with our manager and the client for a project definition review. Following our meeting with GB, our team discussed scope and feasible solutions and assigned duties to individual team members.

Problems Addressed This Week

This week our team faced two obstacles, a team member's illness threatened to interfere with our client meeting and uncertainty about what the client wants. Our initial client meeting was held as planned after the health of our team member improved. At the meeting, we were able to clarify the project scope. The team will address only rental, return, and member processes. All purchasing, inventory, accounting, and billing activities are separate systems. The project remains on schedule as set forth in the attached plan.

Plans for Next Week

The team expects to complete the draft SOW by next Friday and to have an initial draft of the Project Definition Report. We have scheduled a meeting with the client for 4:00 p.m. on 30-Jan to go over any questions that arise. The team also will prepare a first version of the project definition presentation.

Individual Effort for the Week of 19-Jan

Team Member	Activities	Hours This Week	Total Project Hours
Al Price	Attended group, manager, and client meetings. Created weekly report, turned in the team contract, studied tasks and deliverables, assigned work to team members and updated schedule.	8.0	15.0
Dick Von Kemp	Attended group and client meetings, started the narrative, EDM and DFD for project definition.	5.5	9.0
Dan Cartperson	Attended group, manager, and client meetings, started a first draft of the SOW.	7.0	11.0
Terrie Shaftcart	Attended group and client meetings, wrote up client request per group discussion, wrote up two feasible solution classes—build and buy.	6.0	12.0
TOTALS		26.5	47.0

Updated schedule as of 24-Jan

Milestone or Activity	Est. Per. hours	People Assigned	Sequence PreReqs	Start Date	Draft Date	Due Date	Status
1 Organization Plan				13 Jan	19 Jan	21 Jan	Complete
1.1 Deliverables							
1.1.1 Team Contract	10	All		13 Jan	19 Jan	21 Jan	Handed in
1.1.2 Skill Inventory	5	All		13 Jan	17 Jan	21 Jan	Handed in
1.2 Meet with Manager	6	Terrie/Dan	1.1	19 Jan	20 Jan	21 Jan	Done
1.3 Meet with Client	18	All	1.2	13 Jan	23 Jan	24 Jan	Done
2 Project Definition				24 Jan	31 Jan	13 Feb	In-progress
2.1 Review client request	6	All	1.3	24 Jan	24 Jan	24 Jan	Done
2.2 Draft SOW	10	Dan/Al	2.1	24 Jan	31 Jan	3 Feb	Started
2.3 Draft Project Def. Rpt.	40	Dick/Al/Dan	1.3	24 Jan	3 Feb	13 Feb	Started
2.4 Proj. Def. Presentation	30	All	2.2, 2.3	26 Jan	4 Feb	6 Feb	Started
3 Proposed System				1 Feb	28 Feb	28 Feb	Not started
3.1 Prepare specifications							Not started
3.2 Draw models							Not started
3.3 Refine Alternatives							Not started
3.4 Conduct Evaluation							Not started
3.5 Select recommendation							Not started
4 System Acquisition				1 Mar	15 Mar	15 Mar	Not started
Option A. Purchase							
Option B. Build							
5 System Delivery				15 Mar	8 Apr	8 Apr	Not started
5.1 Acceptance Test							Not started
5.2 Implementation							Not started
6 Final Report and Presentation				5 Mar	18 Apr	23 Apr	Not started

will contain only parts of the section I through V materials. A wise and effective analyst will write each interim report so that it can become a part of a final report with only minor editing. Interim reports might include a project definition report, a proposed system specifications report, and a system design report.

Reports may vary in length depending on the project. A good report communicates the message that the team explored the issues in depth and wrote enough about them to clearly and completely describe the issues. A one- or two-sentence project statement may suffice. A one-page project definition or proposed system specifications section suggests to the client (and to the instructor for students) that

FIGURE 3.6 Contents of a Final Report

Title Page
Table of Contents
Executive Summary
Introduction
Part I. Project Definition
Part II. Proposed System Specifications
Part III. Alternatives, Evaluation, and Recommendation
Part IV. Design Specifications and/or RFP
Part V. Implementation and Support
Appendixes

the team did not take the task seriously. The major sections of the report probably will cover from 3 to as many as 10 or more pages.

A final report that contains graphical process and data models for the current and proposed system probably consists of at least 20 pages, and more likely 25 pages or more, plus appendixes. In IT work, good final reports almost always have a number of appendixes. On the other hand, a report of 50 or more pages plus appendixes for the typical small project probably contains a lot of material that is not very useful.

Table of Contents

The **table of contents** lists each major section and appendix of the report through perhaps the second or third level of headings followed by the page number on which the section appears in the final report. For whatever level the team picks, the table must show every heading at that level with the corresponding page number. The type font used in the table, whether Ariel, Times New Roman, or other must match the font used in the report for the headings. When a report is complete, check again that the table correctly lists all of the headings with correct page numbers.

Headings and Fonts

Many heading schemes will work as long as you use a consistent one. For example, consider the following scheme:

Level One

[Place text here.]

Level Two

[Place text here.]

Level Three

[Place text here.]

Use the same heading scheme, typeface, and font size conventions throughout the report. Level one designates the major sections, such as the table of contents, executive summary, project definition, and so on. Level two designates subtopics in level one sections and level three designates subtopics in level two sections. Most writing books ask you to include at least two subheads within a section if you use any at all. If you wish, you can use a different type font or size for the headings. Most PC word processing programs come with a built-in heading scheme. Using a default scheme saves time and effort. Adding complexity to the heading scheme or to any part of the report makes rewriting more difficult and increases the likelihood of making mistakes and inconsistencies. Rapid development focuses on eliminating complexity for this reason.

Executive Summary

The **executive summary** consists of *one or several pages* of text intended for senior managers and clients. A manager or client who reads the summary should receive information on all of the key contents in the report—especially a project statement, strategic alignment, desired features, constraints, specifications, alternatives, evaluation, and conclusions or recommendations. The summary highlights every main point in the entire report and must not introduce any material not in the report. If the team discovers a new point that should go in the summary, the team must first cover the point in the body of the report.

One good approach to preparing a summary is to copy sentences or paragraphs from the body of the report that cover all of the main or key points for the report and paste them into the draft summary document. After editing to remove duplication, examples, and other unneeded materials and to provide coherence and smooth transitions, the draft becomes the summary. The team also may wish to rearrange the order in which points appear. This approach guarantees that the summary actually summarizes materials in the report and does not introduce new content, observations, or conclusions that do not appear in the report.

Introduction

The **introduction** identifies and briefly describes the pieces and parts that make up the report. (The executive summary, by way of contrast, summarizes the most important content or ideas and conclusions from the report.) The introduction offers the reader an annotated version of the table of contents with only the most significant or interesting items described. For example, the introduction probably says nothing about the table of contents or the executive summary but may describe the contents of the current situation and other key sections in some detail.

Good Writing

All reports should follow **good writing** standards. You may wish to review any previous business or general writing courses. If you learned nothing from them, then learn how to write well now. Some examples of writing problems and better solutions appear in Table 3.3.

TABLE 3.3
Examples of
Writing

Questionable	Better
[Past, passive with indefinite pronoun] It has been recommended by the team to use a package program.	The team recommends a package program.
[Weak verb] Direct implementation is the best choice.	Direct implementation offers the best choice.
[Ping-pong] Figure 1 contains the DFD. Figure 2 contains the EDM. The DFD covers three pages.	Figure 1 contains the DFD. The DFD covers three pages. Figure 2 contains the EDM.
[Poor lead sentence] The team recommends alternative 2. The narrative for the current situation outlines two problems with the current situation. These problems relate to the lack of a database and to . . .	The narrative for the current situation outlines two problems. These problems relate to the lack of a database and to . . . [new paragraph] The team recommends . . .
[Indefinite pronouns, poor verb forms] Alternative 3 may be best. This is our recommendation. It was given the highest score by our evaluation model.	The team recommends Alternative 3. This alternative (not just "This") receives the highest score in our evaluation model.

A few suggestions to deal with the most commonly observed problems follow:

1. Use active, present-tense verbs whenever appropriate. "The team recommends . . . The client wants . . . DFDs show . . ." Sometimes transitive verbs work well. "The goal is . . .", but most of the time, active, present-tense verbs work best.
2. Replace indefinite pronouns, for example, "this, it, they" with more definite nouns or phrases. Use "This experience explains the teams . . ." in place of "This explains the teams . . ."
3. Avoid ping-pong paragraphs or sections. Once you raise a subject, finish discussing it before going on to a new topic.
4. The first sentence of a paragraph should relate to the main content of the paragraph.
5. Long, convoluted sentences like this one may fit in some places, but when used often in your deliverables and other documents, for example, a memo, can lead to reader confusion in some, if not many, cases and can even antagonize your reader or, perhaps, your manager who may retaliate by giving you a lower evaluation or salary, a situation you probably want to avoid.
6. Use as few words as possible to express your ideas clearly.

When your report contains good writing, perform a final spelling, punctuation, and meaning check. Reports containing misspelled words that are detectable by a spell-check program indicate an inexcusable lack of care by the team. Spell-check programs catch many spelling problems, but do not catch correctly spelled words used incorrectly, for example, the use of *you* for *your* or *teem* for *team*. Your

team will need one or several good editors to conduct a final review. Finally, check as a group to confirm that the meaning imparted by the report to the reader matches the ideas you intended to convey. A well-prepared report may not save a poor project, but a poor report can damage a good project.

Report Appearance

The client will appreciate a clean, clear, easy-to-read report. A word processing program with an attractive, professional-looking font and a large, legible type size—11 or 12 point for text—will produce a good-looking report. Replace any smudged or visually impaired pages with clean ones. Covers and some form of binding give your report an official look and will protect it. A stapled report that comes apart on first inspection may give your client a poor impression of your work and the pride you take in it. The front cover should display the name of the client's organization, the project name, a date and, if desired, the team members' names.

Presentations

Presentations, for example, a project definition or a final presentation, give the team a structured opportunity to demonstrate their competence to their manager and to the client. Sometimes the client will invite senior managers to the presentation—the CIO or even the CEO. Frequently, the CIO, CEO, and other senior managers will glance only briefly if at all at the written report. However, these senior managers will listen to you. In a world where everyone rushes around with little time to focus on any one issue, a presentation offers a unique environment, a few minutes of relatively undivided attention. If the team does an outstanding job with a presentation, the team can create a long-lasting, positive impression of grace, competence, and spirit. On the downside, a disorganized presentation or a demonstration that fails with no backup can create a negative feeling. The team certainly faces strong incentives to do a good job.

Good presentations do not just happen; preparation counts. A presentation works best when the team produces a good product. The product comes across best when embedded in an attractive, smooth, clear audio-visual (A/V) format presented by alert, clear-speaking, and well-rehearsed team members. In short, a good 30-minute presentation flows from hours, weeks, or even months of hard work.

Guidelines for preparing for a formal presentation include the following:

- Identify strong and significant content for the preparation. Tailor the content to the client; put an emphasis on those things of most interest to the client and your manager.
- Produce appropriate visual aids to help communicate the content.
- Prepare handouts for backup in the event of an A/V failure or as the main A/V media and/or to cover any materials that the clients may find difficult to read in the main A/V.
- Assign roles of who will do what at the presentation.

- Rehearse, revise, and rehearse again until the presentation is smooth and convincing.
- Before the presentation, if possible, visit the meeting room to note and test seating arrangements, existing visual aids, electrical outlets, network outlet, and such and also learn the names of the clients who will attend.
- Make certain that all the team members and the manager know where to go for the presentation and how to get there. Use e-mail or another form of written directions.

Team Member Roles at the Presentation

The team assigns **team member roles** for every member prior to the presentation. Possible roles include

- Anchor—who opens and closes the presentation.
- Presenter—who presents part of the material.
- Technology driver—who operates the projector, runs the prototype, etc.
- Note taker—who records client comments during the presentation; may prepare a summary for the end.
- Host—who welcomes the clients, gives them name tags, shows them to a seat, and converses with them until the presentation starts. The host may stand outside the door to show attendees the way to the presentation.

One person may play several roles, but every team member should perform in at least one role. When several people are going to speak, each speaker can introduce the next. Alternatively, the anchor can introduce every presenter at the beginning or can refer to the printed agenda, which shows who will present what. After the anchor's remarks, each presenter, when his or her turn comes, can state his or her name and start talking.

If the team uses a computer to run a demonstration or to project images, the technology driver operates the computer for the slides or runs the demo while the presenters speak. One person doing both roles tends to slow the tempo and bore the attendees. You can make better eye contact with the attendees while looking at them rather than while looking at the screen. If (when) a computer problem occurs, the driver can try to fix it while the presenter improvises, that is, goes on to the next topic, relates preplanned materials that the presenter originally cut out for lack of time, or if all else fails, invents interesting and useful thoughts. Being a lucid speaker while trying to fix the blasted computer challenges even the most gifted team member.

Visual Aids

Even spellbinding orators benefit from good visual aids, such as colors, pictures, and images of text. The best visual aids to use depend on the purpose and nature of the presentation. For example, flip charts work well at the project definition presentation because the team wants to encourage a high level of interaction with the clients. Computer-generated slides look like a finished product and may inhibit suggestions for changes or additions. For a final presentation, PowerPoint

or similar computer-generated aids allow the team to integrate the slides with the proof of concept model demonstration. IT people should use computer-based technologies. Below are some guidelines for visual aids.

- Select high contrast colors for text and background; use black on a very light color, for example. In general, color adds to a presentation, but slightly restrained colors tend to wear better than a neon orange background with passion purple and fluorescent green letters. The goal is to sell, not to nauseate, the client.
- All type must appear in a large enough font to be read easily at the back of the room. Copies of documents or tables with a lot of text and numbers seldom show up clearly on a screen. Consider using handouts for these kinds of displays.
- Half or more of the charts used in your presentation should have specific content that you want the client to note and/or to remember, such as data, a list of benefits, or other important information, not just an outline of what you will say. An occasional outline chart is fine, but make sure the content ones are there as well.
- Use sounds and special slide transitions with caution. Noises and pictures fading in and out, flying in from an edge, or flashing around on the screen may distract the client from the content. Avoid automatic timing such as setting the projector to display a new slide every 30 seconds.
- Look carefully at your proof of concept visuals. You may wish to include screen shots, input forms, or reports with content that is difficult to read. If so, prepare a handout.

The content on the visual aids will vary by project and client. Typical visual aids for a final presentation on a project may include:

- A *title slide* that shows the project name, client's organization, and client(s) names(s). Clients like to see their names appear. The slide also may include the team members' names.
- A *project statement* that summarizes what the client asked the team to do.
- An *agenda* for the presentation.
- *Strategic alignment* which answers how will the solution contribute to the organization's goals.
- *Features* for the proposed solution. What features does the client want or ask for?
- *Constraints* that the client placed on the proposed solution.
- *Specifications* for the proposed solution. Use one or more slides on the specifications that the team developed based on the client's desired features and constraints.
- *Alternatives* provide one or more slides for each of the alternatives with a description, advantages, disadvantages, costs, benefits, risks, and so on.
- *Evaluation summary* includes a table that shows the key criteria for alternatives with a value for each criteria for each alternative that ties to strategic alignment.

- *Recommendation* gives the selected alternative and the key reasons for selection. This slide should tie back to the evaluation summary and to the strategic alignment.
- *Introduction to the proof of concept (POC) demonstration.* Include a slide or slides that cover the key features and limitations of the proof of concept model. Is it a prototype or package? What features are missing? What are limitations of the data in the tables? Most clients want only an overview, not the full technical specifications.
- *The POC demonstration* uses live operation, if possible. If live operation is not possible, use screen shots of inputs and outputs.
- *Implementation issues* include approach, schedule, players, testing, maintenance, and more.
- *Summary* should restate the recommendation and any other key issues. Finally, thank the client(s) for their time.

Some presentations may contain all of the above slides and more. Others may contain only selected slides from the above list plus any needed additions. The team may wish to change the order, for example, to cover implementation before the demo.

Rehearsal

Even the best team can benefit from **rehearsals**. With enough rehearsal, the presenters can appear comfortable and spontaneous while still covering all the points in the allocated time. Experiment at the rehearsal with the proper role for each presenter and how to transition from one person to the next. Check carefully the readability of your visual aids. Can you hear and understand the speaker? Constructive criticism at this point can lead to a much stronger final version. Keep a record of times. The actual presentation probably will run longer than the rehearsal: Extra comments or a little confusion and repeated sentences tend to creep into the final presentation.

During the rehearsal, plan for contingencies. The Basic Rule of Presentations holds that what you least expect or most fear may happen; imitate a good Boy Scout: “Be prepared.” An unwanted event that you are prepared to handle may cause annoyance. But if you are not prepared, it escalates into a crisis or a disaster. For example, plan what the team will do if the following interruptions occur.

1. A team member becomes ill and/or does not show or must leave.
2. A team member becomes confused, stops, or forgets to cover material.
3. The video projector will not work or fails in the middle of the presentation, or the room is too light to see the images.
4. The proof of concept demonstration does not work.
5. The client asks so many questions that the team cannot finish.
6. The client strongly objects to or argues with some of your material.
7. The CEO or CIO announces that she must leave but will return shortly.

The preceding list gives only a few examples; many other problems can arise.

For the Presentation

The team must prepare and present themselves in a professional and courteous manner. The team will need to be aware of the following suggestions:

1. Wear appropriate clothes.
2. Know where you are going and how to get there.
3. Plan to arrive far enough in advance to allow for bad traffic, minor mechanical failures, getting lost, waiting for an escort to enter the building, or other potential delays.
4. Set everything up and check for proper operation before the clients arrive.
5. With the client's permission, set the room up the way you want it. Do not feel constrained by an existing bad arrangement. Set up the room so that all the clients can see and read the visual aids with comfort. If possible, arrange seats so that each person can see all of the others.
6. Position a team member in the hall outside the room to welcome attendees and help them find the place.
7. Greet every attendee, give him or her a name tag, and help him or her find a seat.
8. Bring name tags and/or place cards for all the clients and the team members. Bring blank extras in the event unexpected clients arrive.
9. Start on time or as soon as the key clients arrive and agree to start.

During the Presentation

Presentation and professionalism are the key to success of any presentation. The team should always be aware of the following concepts:

1. The team members may stand at the front of the room during the presentation or may sit down when not speaking. If the team members are seated, intermix team members and clients in the seating; avoid an "us" versus "them" seating across a table that may seem adversarial to the client.
2. Welcome the attendees and thank them for coming.
3. Either introduce everyone or ask each person to introduce him- or herself.
4. Ask permission to record, if relevant.
5. Indicate how you plan to deal with questions—during or at the end or both.
6. Apply what you learned during rehearsals to realize a smooth, exciting, and interesting presentation.
7. Speak loudly enough for the clients to hear without strain. If the room has a noisy air conditioner, you may need to speak quite loudly. Watch for any signs that the clients are straining to hear. Ask them if they can hear. Shout if you must.
8. With small groups (up to about 5 or 6 clients), welcome and introduce a client who arrives late or ask the client to introduce him- or herself. With larger groups and people who straggle in after the presentation starts, the

team may choose to recognize the arrival of only the most key clients: CIOs, CEOs, and such.

9. Encourage interaction. Ask questions if the clients do not volunteer comments. During the discussion, follow the guides for interviews: avoid leading questions, do not interrupt the client, and so forth.
10. Restore the arrangement of the room and clean up any mess you created before you leave.

The Final Presentation

The team prepares for the **final presentation** throughout the project. All of the work during each stage of the project contributes to the final presentation. Follow the general guidelines for presentations. Normally, the team will use visual aids, computer-generated visuals if possible, for the final presentation. The content of the final presentation will vary depending on the nature of the project and client. General content guidelines appear below.

1. A brief description of the project—what the client asked the team to do.
2. The agenda—what the team plans to do for the presentation.
3. A brief, concise summary of the strategic alignment for the project.
4. A review of goals or requirements and constraints for the proposed system as defined by the client.
5. A summary of the specifications developed by the team for the proposed system.
6. A discussion of any special activities the team undertook to find solutions.
7. A review of alternatives the team considered.
8. A presentation of the evaluation of alternatives. The evaluation must relate to and build on the strategic alignment for the project.
9. A recommendation that is clear, strong, and unambiguous. Earlier in the project, the team discussed the recommendation with the client and received client approval to proceed. The client does not want to hear any uncertainty or lack of enthusiasm at this point, especially in front of the CIO, CEO, or other officer.
10. Demonstration of the proof of concept model for the recommended system.
11. Implementation and maintenance plans.
12. Summary including a strong restatement of the recommendation.

The amount of emphasis and time that the team devotes to each topic will depend on previous interaction with the client(s). If all of the clients present are familiar with some of the content areas, the team can mention or briefly cover the familiar areas and spend most of the time on the less familiar areas. If a high-level client with little previous exposure to the work attends, the team should cover carefully the areas of most interest to the senior person, for example, strategic alignment, evaluation, and recommendation. If the client attendees are technical types, the team may spend more time on specifications, alternatives, build or procure issues, and implementation.

Most teams ask the client to schedule one hour for a presentation. A busy or impatient client, however, may limit the team to 30 minutes. The team's presentation including demonstration of the proof of concept model should not last longer than two-thirds of the time the client has allocated; for example, in a typical one-hour slot, allow no more than 40 minutes of presentation in order to leave at least 20 minutes for discussion. Many teams encourage the clients to ask questions as the presentation proceeds and the client's questions may use up part of the planned discussion time. The team needs to make sure to present the most important points before the client leaves. If due to questions, the presentation is running longer than planned, ask the client if more time is available. If an important client must leave, try to stop any discussion, make the critical points, and then resume discussion.

Summary

Project management offers the team tools for effective action in four areas: (1) decide the tasks that are required to complete the project, when to perform each task, and the role of each person in the project; (2) convert plans into action; (3) monitor progress and take action as required to deliver the product on time and on budget; and (4) use oral, visual, and written media to inform clients and managers about progress and results. The first area addresses *planning*, the second *control*, the third, *execution*, and the fourth *communication*. In addition to such direct activities as project definition, proposed system specification, system design, and system delivery, an effective analyst participates in a number of project management activities.

Planning represents a fundamental project management activity. With good project management, planning continues every day throughout the life of the project. An SDLC plan typically consists of a sequence of formal steps that may begin with organizing the team and project and continue to the delivery and support of a new system. The spiral model for project planning extends the concepts of the SDLC model to recognize that project activities often follow a cyclical path. Modern systems can benefit from a compromise plan that retains some of the structure of the SDLC/spiral models with the flexibility to produce results rapidly at an affordable cost. The new rapid development planning model selects parts from and assigns priorities to the SDLC/spiral tasks. Rapid development represents a highly flexible concept: "Do only what is necessary to deliver an application that (1) meets the client's perceived needs, (2) in as short a time as possible, and (3) at the lowest possible cost." Analysts achieve these RD goals by incorporating one or more of the following concepts into the project plan:

- Include only tasks or activities that are essential.
- Use programming tools and/or languages that allow fast development and easy changes.
- Work with planning mechanisms that facilitate communication and rapid feedback between analysts and clients.

The “rapid” piece of rapid development holds special importance. Clients want new applications tomorrow not in two years. Speed of development also can improve the chance of success and reduce cost. Scope creep or the addition of features to a system after the initial requirements are set, happens when clients change their minds about requirements or reorganization brings a new set of requirements. The problems and costs associated with changing clients, requirements, and developers are minimized by plans with short development times. Even with RD, a key project management task is to control client changes in requirements without antagonizing the client. If the team can demonstrate the cost of a change to the client, such as a delay in completion time or the risk that the project will not succeed, then the client can make a rational decision on whether or not to proceed with the change.

Teams can choose from a number of mechanisms that facilitate preparation of a rapid development plan. Under the contract approach to RD plans, the client and team, or perhaps a manager or CIO, jointly agree on a broad plan, including the deliverables, major tasks, and schedule for the project. The statement of work (SOW) represents one approach to a contract-based rapid development plan. The joint team approach to RD planning involves forming a team composed of analysts, clients, users, and perhaps others such as managers, to work together intensively from start to finish on the project. In place of a detailed predetermined plan, prototype-based plans start by building an initial physical version of the application and use the prototype to answer some of the project definition, proposed system, and system delivery questions for the new system.

Increasingly, organizations acquire part or even most of their IT capability through some form of outsourcing, which is purchasing an application or IT service from a vendor. Outsourcing represents another approach to rapid development, a way of trying to reduce time and cost for the development process. Purchasing a packaged application or an IT service represents a common form of outsourcing. The team also can evaluate the outsourcing of IT functions by finding an external organization to do some or all of the development or to perform the functions of the system. A common way to outsource a project is to use a request for proposal (RFP). To respond to an RFP, the vendor, in effect, performs parts of the system creation process.

A project plan requires several steps: (1) identify the appropriate activities—task decomposition; (2) estimate person hours and sequence constraints for each activity; and (3) build or generate the schedule in a format that managers and team members can understand. While all plans follow this basic structure, the rapid development philosophy means that each system plan will evolve differently as a function of the requirements for each specific project. A complex system may require the use of a project management tool to build the schedule and identify the critical path. For moderate-size projects, an activity table or a Gantt chart offers helpful graphical model representations for the schedule.

Project execution and control, two closely interrelated functions, start with the process of converting plans into action, that is, they put team members to work on the various activities needed to meet the plan. Project execution and

control operate in the context of a feedback control system; they take initial action, observe results, compare to plan, take action as needed, observe results, and more. The team uses the project control activities to identify problems and opportunities. The team can then take new actions to alleviate the problems, capitalize on the opportunities, and use the project control function to monitor the results.

Corrective action represents a most difficult aspect of project management. At every team meeting, the team should discuss not only status, but also the actions that might correct any problems. Possible actions to correct problems that arise when actual progress falls behind planned progress include: change the expected completion date, reallocate resources, work more effectively, work harder, find a consultant, and/or reduce the project scope. When a problem first occurs, a number of the aforementioned actions may work. A month later, none of them may work. A key to effective correction action is to take action as soon as the problem is detected. Although a team can and should take action any time a problem appears, most organizations also identify and conduct in-process reviews at key points during the project.

Project management involves a number of time-consuming jobs. Fortunately, computer-based project management tools can perform many of the routine tasks. These tools provide convenient formats for input data on tasks, durations, dependencies, and resources; create an initial schedule; update the schedule in response to change data; and generate a variety of analysis and reports. Teams can choose from literally hundreds of project management tools. The tools require an initial purchase and, more important, an investment by users in learning how to use the tool. Industry surveys suggest that Microsoft Project is the most widely used tool, but a number of others also are used.

Communication with team members, managers, and clients forms an essential part of every project. The team can deliver routine messages and reports by mail, e-mail, or telephone; the delivery of significant bad news works best with a face-to-face visit. Effective management of a project requires good information on progress. Normally, progress monitoring on a weekly or longer basis can provide all the information a manager can use. A typical weekly report contains a summary of significant team accomplishments, problems and corrective actions; a time or effort report by task and person; and an updated RD plan.

The team may deliver interim and final written reports and presentations. The reports and presentations help the client to evaluate the team's work and provide a base for any additional or future work including system operation, maintenance, and modification. The contents and sequence of the reports and presentations follow the organization's standards, if standards exist. A standard model starts with a description of the problem that the client asked the team to address, describes the team's accomplishments to date, and ends with work required in the future. Presentations give the team a face-to-face opportunity to demonstrate their competence and sell their product to the client. A good product comes across best when embedded in an attractive, smooth, clear audio-visual format presented by alert, clear-speaking, well-rehearsed team members.

Key Terms	activity schedule table, 79	PERT chart, 91	reviews, 89
	CASE (computer-aided software engineering), 67	progress report, 91	scope creep, 68
	client/team contract approach, 69	project communication, 91	SDLC (systems development life cycle), 65
	corrective action, 88	project control, 84	sequence constraints, 76
	evolutionary prototype, 72	project execution, 84	spiral model, 66
	executive summary, 96	project management (PM), 64	statement of work (SOW), 80
	final presentation, 103	project management tools, 90	table of contents, 95
	Gantt chart schedule, 78	project planning, 64	task times, 76
	good writing, 96	project schedule, 81	team member roles, 99
	introduction, 96	proof of concept, 72	throwaway prototype, 71
	joint team approach, 70	prototyping, 70	weekly report, 92
	monitor progress, 87	rapid development (RD) planning model, 68	work product, 80
	outsourcing, 72	rehearsal, 101	
	packaged application, 73	resistance to change, 85	

Review Questions

1. What is the definition of project management?
2. What are the main activities or subtopics of project management?
3. How can an analyst achieve RD goals?
4. What are the differences between the SDLC and the spiral models?
5. What are some of the conditions that make joint client/analyst design sessions particularly useful?
6. In what situations does prototyping seem to work the best?
7. Why would an analyst want to design a throwaway prototype?
8. What is meant when one uses the phrase, "Plans should evolve over time"?
9. What are the main components of a SOW?
10. Since IT projects are about change, why do users resist change?
11. The project is falling behind because of a bug in the code that the team cannot correct. What should the team do?
12. Describe the guidelines for good visual aids to be used in a presentation.
13. Why is communication important for project management?
14. You have been chosen to send the weekly progress report to your manager. Design the format of the report and add data to show a sample first week report.

Critical Thinking Exercises

Individual Exercises

1. You have been asked to develop a client database for a marketing organization. There are several users in various locations throughout the state. Your manager has asked you for your advice regarding a rapid development approach. What is your recommendation and justification for your approach?
2. A local business wants you to develop a simple informational Web site in HTML. The business wants the site to reflect the business goals, the product line, the owners, business hours, and the location. The owners want pictures of themselves and the location on the site. You were advised in your systems class to get everything in writing before

beginning the coding. Prepare a statement of work that will protect you and the business owners.

3. For exercise 2, describe how you will determine what to charge the business owners for your work. Include your sources in your answer.
4. Again, using the information shown above in question 2, you have been working on the project for several weeks and are ready to present it to your client. This morning you received a call from the client wanting to add several pages to the site. How will you approach this problem with your client?
5. You are a member of a team that has worked hard and prepared well for your final presentation. What will you do if the following situations occur?
 - a. A team member with a crucial role in the presentation becomes ill and cannot show up for the meeting.
 - b. The client continues to interrupt and your scheduled time frame is close to being over.
 - c. The client strongly objects to your proof of concept model.
 - d. One of your team members constantly interrupts your client.

Group Exercises

1. Your team has agreed to arrive at project definition review presentation with the client wearing business casual clothing. The team's definition of business casual is slacks, sports shirt, dress shoes for the men and skirt and blouse with dress shoes for the women. One of the team members arrives in blue jeans, sandals, no socks, and a pullover T-shirt with crude lettering on the back.
 - a. What should the team do if you discover the member outside the client's office?
 - b. What can the team do if the offending member arrives in the room with the client?
 - c. How can a team prevent this problem from happening?

The following questions use mini-cases that appear in additional questions in later chapters.

2. Your team has been given the assignment of building or buying a package to track ordering and issuing of office supplies for a major law firm. The senior partner is your primary contact. She has the impression that supplies seem to walk out of the office, especially as each school year starts. She wants to set up a system that will provide controls on issues, reports on usage by division, and procurement based on need rather than when a bin is empty. She wants the system to assist in an annual physical inventory, which she believes will minimize pilfering. Your team is to do the following:
 - a. Set up a schedule of activities for the project in activity table format.
 - b. Set up the schedule in a Gantt chart.
 - c. For each important piece of information in the schedule and plan, evaluate the effectiveness of an activity table versus a Gantt chart for displaying the information.
 - d. Prepare a statement of work for this project.

Your team must also answer the following questions:

- e. How will the team members communicate with each other?
- f. How will the team monitor progress against the plan?
- g. What will you do if progress as shown on your Gantt chart falls behind your planned progress?

3. The Business Association is conducting a babysitter service as a fundraiser for different clubs in the college. When a customer is entered into the system, the Association coordinator gets the customer's name, address, and phone number. The coordinator also records each babysitting job, the amount paid for it, and the sitter assigned to the job. Each person may sign up to credit only one club, and the system keeps the contact person and phone number for each participating club.

The treasurer wants to determine how much each customer was billed by week, month, or year and how much each employee earned, also summed by time periods. The treasurer is interested in how much work is done on weekends, holidays, or other special days.

- a. Set up a schedule of activities for the project in activity table format.
 - b. The team has three members, Al, Jane, and Kim. Assign each activity on the chart to one or more of the team members.
 - c. Explain why activities were assigned to team members. To answer, make up skills for Al, Jane, and Kim.
 - d. The team is far behind schedule, but the advertisements distributed all around the campus give next week as the starting date for the service. What should the team do?
4. The Motor Vehicle Pool (MVP) at the University of Oklahoma rents vehicles to university departments. A vehicle may be rented for short trips or for long-term use. University departments renting a vehicle must have an employee who is a certified driver to operate the vehicle. Payment is not made at the time of the rental; rather, payments for all rentals are made at the end of each month. On the last working day of each month, the funds for each department's rentals are transferred to the MVP operating account from each university department that rented vehicles.

The MVP has a total of five employees. Two of these are office clerks, who take reservations, process paperwork, and update vehicle cards. These two employees are each paid \$20,000 per year with 33 percent benefits. The MVP has two mechanics who provide minor maintenance on the vehicles such as tune-ups, hose replacement, belt replacement, and other minor repairs. These two persons are each paid \$26,000 per year with 33 percent benefits. All major maintenance and bodywork is contracted to the lowest bidder. The bidding is based on the hourly labor rate and the parts are bid at cost plus their profit. These contracts are awarded annually to begin on January 1 and extend through December 31 of any given year. Finally, the MVP has a pool manager, who is responsible for the MVP operation, ordering new vehicles, and setting up the annual auction of vehicles to be retired. The pool manager is paid \$50,000 per year with 33 percent benefits. Each vehicle is retired and auctioned after four years of service or 200,000 miles, whichever comes first.

When a university employee wishes to rent a vehicle, they send an e-mail or make a phone call, requesting a specific type of vehicle for a specified time frame. The MVP clerk who makes the reservation checks to see if that type of vehicle is available for the requested time, and if the person who will be responsible for the vehicle has the appropriate license. The MVP manager must certify each driver of a university vehicle. If the vehicle being requested is designed to carry 15 or more passengers, the driver must carry a class C license. All other university vehicles that can be rented require only a class D or regular driver's license. Before a driver can be certified as a qualified university driver, the driver candidate must pass a written and driving test for the specific vehicle type, which will be administered by the MVP manager. It does not matter what civilian state driver's license one holds, the potential university driver must be certified by the university.

If the requested vehicle is available, and the driver/operator has been certified by the university, then a reservation is recorded on the vehicle card. When the vehicle is turned over to the driver, any dents or major scratches are duly noted on the rental form. The driver personally inspects the vehicle and indicates that he/she agrees with the MVP clerk on the condition of the vehicle. While the university is self-insured, the MVP manager suggests that the driver also be personally insured by his/her own insurance company. The vehicle is then checked out with a full tank of gas, and the renter told that the vehicle needs to be returned with a full tank. If it is not, the department will be charged the current price of gas at the ConocoPhillips station, plus a \$10.00 surcharge to cover the maintenance personnel's time to obtain the vehicle's fuel.

At the time of the rental, the following information is recorded about each driver on a rental card: university ID number, civilian driver's license number, university driver's certified number, driver ID data (e.g., hair color, eye color, weight, height, driving restrictions), department account number, department name, job title, years with the university, and renter's driving record (e.g., accidents, driving violations, etc). This is self-reported, and each time an authorized operator rents a vehicle he/she is asked to update his/her driving history. There is a no-tolerance policy on this issue, and an employee will be terminated for failure to report accidents and/or driving violations.

Additionally, the following information is recorded about the vehicle on the card: the vehicle tag number, VIN, passengers, mileage at last check-in, outgoing condition, maintenance record, mileage at time tires were replaced, record of maintenance activity, where work was done, what was done, cost of maintenance for this action, mileage at time of maintenance, record of use, driver, reason for use, beginning mileage, ending mileage, list of other university personnel (name and ID) on the trip with vehicle (may be in another vehicle or traveling by an alternate mode of transport), and vehicle accessories such as snow tires or cell phone hookup.

The cost charged to the renting department is based on the cost of operating the MVP, prorated to each vehicle. The cost of the MVP department is all operating expenses and vehicle depreciation.

When a vehicle reaches 200,000 miles or is four years old, the MVP manager will set up an auction. This is only done once a year in the late spring, so some of the vehicles may be a little older than four years, and some may have a few more than 200,000 miles. The auction is held at the motor pool yard and is an open bid procedure. The funds received are deposited in the MVP operating account.

At various times during the year, the MVP manager orders replacement vehicles based on demand and forecast of needs. The new vehicles are purchased by the university general operating fund, and set up so that depreciation can be deducted as expenses to the MVP operating account. The information on new vehicles is forwarded to the MVP from the university purchasing department.

The beginning of a school year brings a heavy load of new university personnel who need to be certified by the university as qualified vehicle operators. A group of volunteers who have been previously certified to be university drivers have agreed to function as driver's test examiners when the manager of the MVP is overloaded. Each spring, volunteers are requested from the cadre of qualified drivers and are taught how to administer the driving test. This qualification is noted on the individual's driver information card.

When the driver arrives to pick up the vehicle, he or she is provided with a rental report. This report contains the following information: driver's name, driver's university ID, driver's license number, driver's University Certification Number, destination,

time and date of checkout, anticipated time and date of return, vehicle condition, and other university personnel in vehicle.

At the end of each school year, a summary report of usage by department is prepared for each department director. It contains the same information as the rental report, except that instead of anticipated time and date of return, it has the actual time and date of return. Within each department, the information is organized by driver. An accident/incident report is prepared for each department director as soon as the accident/incident is reported to the MVP manager. An overall vehicle summary report is prepared at the end of each fiscal year. This report summarizes use, maintenance, and accidents by vehicle.

Currently, the MVP manages the vehicles manually. Your task is to specify the requirements to automate these processes. The information provided to you for this case is very realistic, compared to information that you would collect through specification requirements interviews with your client. As you work the case, you will very likely discover the need for additional information or clarification on given points. You will need to clarify with your instructor the procedure to follow when questions arise within your team.

- a. Set up a schedule of activities for the project in an activity schedule table.
- b. Prepare a statement of work for this project.
- c. What skills including knowledge of technologies and organization functions are important for this project?

References

- Boehm, Barry W. "A Spiral Model of Software Development and Enhancement." *IEEE Computer*, May 1988, pp. 5, 61–72.
- Forsberg, Kevin; Hal Mooz; Howard Cotterman; and Norman Augustine. *Visualizing Project Management: A Model for Business and Technical Success*. New York: Wiley, 2000.
- Schwalbe, Kathy. *Information Technology Project Management*. Cambridge, MA: Course Technology, 2000.
- Wood, Jane; and Denise Silver. *Joint Application Design: How to Design Quality Systems in 40% Less Time*. New York: Wiley, 1989.

Chapter Four

Data Modeling

Chapter outline

Introduction

Entity Relationship Data Modeling

Model Components

Entity Relationship Diagram Symbols

Building a Simple ERD

ERD Rules

Basic Rules and Guidelines

Naming Rules

Additional Constructs for ERDs

Multivalued Attributes

Weak Entities

Associative Entities

Degree of a Relationship

Minimum Cardinality

Supertypes and Subtypes

Simplified, Reduced-Form ERDs

Conceptual Data Models

Metadata

Enterprise Data Models

Logical Data Models

The Relational Model

Basic Concepts

Rules for Relational Models

Relational Schema

Converting ERDs to Relational Schema

Unary Relationships

Normalization

Structured Query Language

Dimensional Models

Summary

Key Terms

Review Questions

Critical Thinking Exercises

Individual Exercises

Group Exercises

References

INTRODUCTION

Data and data models affect all aspects of the lives of people. For thousands of years, people have communicated with each other using visual and/or verbal symbols. *Data* are symbols, abstract representations of real things, that convey meaning. These representations include words, signs, writing, sounds, codes, and more. In order to facilitate the transfer of knowledge, data require a structure of rules and conventions. Many information systems provide the structure for collecting, manipulating, storing, and presenting data. In other words, data play a central role in many information systems. Until several decades ago, information systems often were known as data processing systems.

All information systems use a data model of some kind, that is, some specifications or structures for the data included in the system. The analyst may think about the data and incorporate a data structure (a data model) in the program. In COBOL programs, one data model is the Data Division of the program, which provides specifications for all the data used or referenced by the program. Non-IT people often prefer a narrative data model—a text or natural language description of the data in the system.

In 1976, Peter Chen proposed the entity relationship data model, a well-structured data model that was independent of the design and implementation of a current or proposed system. Chen's work was extended and modified by a number of other people. This chapter examines several structured data models. Models provide a standardized representation that facilitates communication between the people working on a project or system. Chapter 1 discusses the role of models and the concepts, advantages, and disadvantages of structure. As noted in Chapter 1, in the content model view, an information system contains data, process, physical infrastructure, and organizational infrastructure. This chapter discusses data models while Chapter 5 discusses process and object models.

Data modeling concepts and tools discussed in this chapter include

- Entity relationship data modeling.
- Conceptual data models (CDMs).
- Enterprise data models (EDMs).
- Logical data models—relational schema.
- Data modeling languages.

Entity relationship diagrams (ERDs), used in Chapters 7 and 8, provide a structured graphical picture of the entity-relationship data model for an organizational area. ERDs come in a variety of formats. The Enterprise Data Model version of an ERD is used in Chapter 7 and the Conceptual Data Model version is used in Chapter 8. Chapter 11 illustrates the application of relational schema.

The chapter strives to cover material that often makes up a demanding one-semester course. The chapter focuses on the topics most important to a person who works on or manages information system projects, but obviously not every relevant aspect of every topic is covered. Most textbooks on systems analysis and design contain a number of chapters on data models, for example, see Whitten, 2005 or Hoffer, 2005a. More in-depth coverage of data models appears in such database texts as Post, 2005 and Hoffer, 2005b. Many books cover principles and programming with SQL; see for example, Pratt, 2003.

ENTITY RELATIONSHIP DATA MODELING

Entity relationship modeling provided a new perspective in which to analyze and design the data structure needed for an organization to carry out its mission. Entity relationship models describe data independently of the way that current or future systems use the data. Analysts often designed traditional data structures in the context of a specific system by thinking about forms and files. Finding an answer to what data were needed for the system involves answering such

questions as (1) What should the input data forms look like? (2) What files are needed? and (3) What reports should the system produce?

The forms shown in Figure 4.1 (from the GB Video example in Chapter 7) describe in a traditional manner the data structure for the GB Video rental activity. These forms include

- *Data Input Form—Member Data Card* (Figure 4.1, Exhibit 1). The customer's name, member number, address, telephone number, credit card number, and other information is entered on an input form. The Customers file stores the data on the Member Data Cards.
- *Data Input Form—Invoice* (Figure 4.1, Exhibit 2). The invoice form contains three types of data: (1) customer data—member number and customer name; (2) rental data—rental number, date, payment type, and amount of payment; and (3) video data—video number, video name, due date, return date and overdue charge. The Invoices file stores the invoice data.
- *Data Input Form—Video Rental Card* (Figure 4.1, Exhibit 3). The form contains two types of data. The video number, title, and vendor data are entered when the video is purchased by the store. The rental number date out and date in data are entered each time the videotape or DVD is rented. The video rental data is stored in the Video Rentals file.
- *Data Output Form—Customer Receipt*. Contains the same data as Invoice.

This example illustrates some possible problems with the traditional data structure created by thinking about the input and output forms used in a current system or application. Programmers often implemented the traditional data model in the logical form of sequential or “flat files” built to meet the needs of the specific application. With flat files developed this way, even minor changes in programs often resulted in a need to restructure the data at considerable cost. A new system may use a different set of forms and files. The traditional structure also may contain duplication of data. For example, the invoice file contains duplicated data from both the customers and video rentals files.

The **entity relationship model (ER model)** addresses the problems of the traditional data model. The ER model ignores the current system and asks instead the following questions:

- What are the entities—the things in the organization or activity under study about which the client wishes to collect and maintain data?
- What are the organizational relationships between the entities about which the client wishes to collect data?
- What are the attributes—the specific items of data the client wishes to collect about each entity?

Model Components

An **entity** is a person, event, object, place, concept, or thing in an organization about which the client wishes to maintain data. Examples of entities include

- Person—customer, student, employee, member.
- Event—rental, sale, repair, enrollment, flight.

FIGURE 4.1
GB Video
Forms

Exhibit 1. Member Data Card

GB Customer Record	
Richard Jazzperson	1346
Name	Member Number
307 Brooks Norman, Oklahoma 73019	VISA 9444 5432 6666 1234 04 09
(405) 325-0768	

Exhibit 2. Invoice (Copy 1) and Customer Receipt (Copy 2)

GB Video Stores					
Date:	11-2-2006	Emp #:	175	RENTAL NO.	1715
Member :	Richard Jazzperson	Member No :	1346		
Video #	Title	Due Date	Cost	Return Date	Overdue Charge
15751378	Patriot Games	11-3	\$ 2.00		
6613498	African Queen	11-4	\$ 3.00		
			\$		
Pay Type : Cash <input checked="" type="checkbox"/>			Tax : \$.40		
Credit <input type="checkbox"/>			Total : \$ 5.40		
THANK YOU					Copy 1 Store

Exhibit 3. Video Rental Card

Video No :	6613489	Title :	African Queen		
Date Acquired	6-1-06	Vendor :	129		
Rental No.	Date Out	Date In	Rental No.	Date Out	Date In
1497	6-3-06	6-4			
1558	8-7-06	8-9			
1579	8-20-06	8-21			
1715	11-2-06				

- Object—videotape or DVD, product, vehicle, house.
- Place—city, zip code, area, store, plant.
- Concept—military unit, work group, bank account.

An entity in entity relationship models represents a **class**, set, or group of things. For example, the person entity, Customer, in GB Video can represent all of the customers for GB Video. An individual customer is referred to as an **instance** of the entity class called Customer. Exhibit 1 in Figure 4.1 shows the data for a customer with the name of Richard Jazzperson, that is, one instance, of the entity Customer. In this chapter and text, the word *entity* always refers to an entity class. Other entities in the GB Video example might include an event entity, Rental, and an object entity, Video.

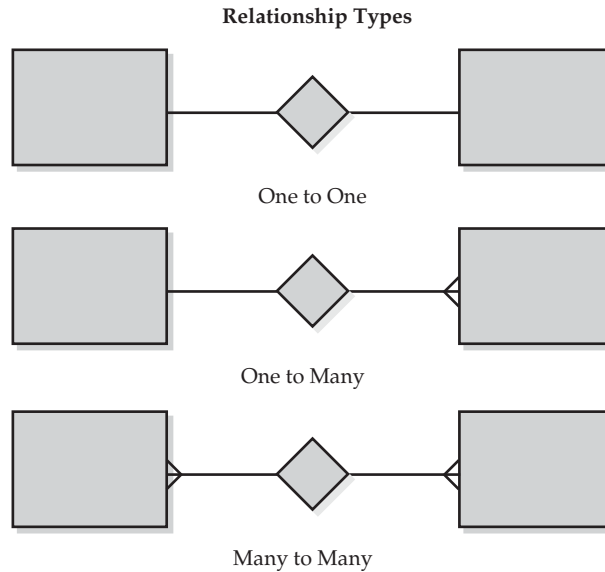
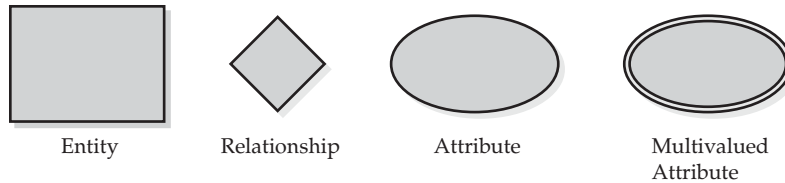
For each entity, the **attributes** of the entity define the properties or characteristics of the entity class about which the client wants to maintain data. Attributes can include keys, names, descriptions, dates, size, color, quantity-on-hand, address, telephone number, and many others. For example, the attributes for the entity Customer as shown in Exhibit 1 of Figure 4.1 include name, member number, address, telephone number, credit card number, and credit card expiration date.

In entity relationship models, every entity must have an attribute or a set of attributes that serve as the **primary key** for the entity. The attribute(s) that serves as the primary key must have unique values over all instances—each value for the primary key can appear only once among all of the instances that belong to the entity class. For example, Member Number probably is a suitable primary key for the entity Customer. Name may not work because two customers may have the same name, for example, two Joe Smiths. The combination of name and address probably will work most of the time as a primary key, but two Joe Smiths might live at the same address. Some text materials use the term *primary identifier* instead of primary key to describe the unique identifier for each instance in an entity.

A **relationship** links or connects instances of one entity to instances of another entity to describe the way the organization functions. Relationships, in common with entities and attributes, are derived by the analyst from studying the way the organization operates. For example, in GB Video, the entity Customer is related to the entity Rental in that a Customer can engage in or make rental transactions. One of the major goals of models is to provide a standardized representation that facilitates communication between systems people working on a project or system. In ER models, the standardization comes from standard symbols for each component and from standard rules to construct a graphical representation of an entity relationship diagram that describes the data model. The next section illustrates standard symbols used in ERDs. The following section describes constructs and rules and gives examples of ERDs.

Entity Relationship Diagram Symbols

Figure 4.2 illustrates one possible set of standard symbols for ERDs. The symbols in Figure 4.2 include only some of the ones used in ERDs. Other symbols will be introduced as needed in the text. In practice, people use a variety of different symbol sets. This chapter will use another symbol set later to illustrate some of

FIGURE 4.2
ERD Basic Symbols

the differences an analyst may encounter. Once the basic ERD ideas are mastered, the analyst should be able to understand any symbol set after a few minutes of study. The next section illustrates how to use the symbols in Figure 4.2 to construct an ERD.

Building a Simple ERD

To build an ERD, the analyst combines the symbols to represent the structure of the data in the organization as it relates to the problem posed by the client. The analyst examines the organization to identify the entities, attributes, and relationships. Some analysts find it useful to write out a description of how the organization works; others, particularly analysts with experience, can just look at or think about the organization and translate it to an ERD.

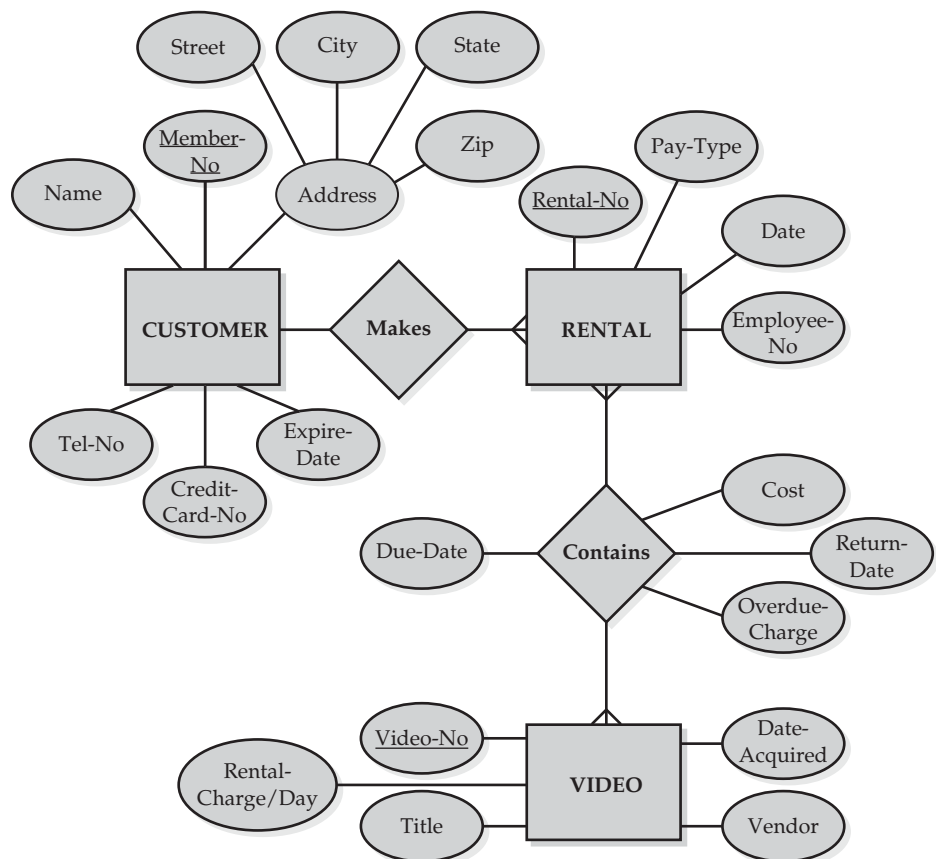
Consider the following simple narrative that describes how an organization rents videos, videotapes or DVDs, using the forms shown in Figure 4.1. The narrative is a simplified version of the GB Video rental process discussed in Chapter 7.

Customers may rent one or more videos, videotapes or DVDs, for one or more days. The customer brings the desired videos to the counter along with his or her membership card. The clerk copies the name and number from the card and copies the title and unique ID number from the label on each video onto a prenumbered form for the rental. If the customer forgets his/her card, the clerk

looks up the number in the customer file. The clerk also enters his/her employee number, the date of rental, and a due date for each video computes the amount of the charge and tax and gives a copy of the form to the customer as a receipt. Customers may pay cash or use a credit card for rentals. The clerk notes the payment type on the form.

Analysts follow different paths to create ERDs for a system. Some analysts like to start by identifying all of the entities since entities form a central focus of ER models, and then add the attributes and relationships. By reading the narrative and thinking about the organization, an analyst could determine that this organizational activity involves a person entity, CUSTOMER and an object entity, VIDEO. Note that each of these entity classes represents a number of instances, such as a number of customers and a number of videos exist. The narrative also describes an event, RENTAL, and a number of instances of rental exist. At this point, the ERD can contain the three entities CUSTOMER, VIDEO, and RENTAL. These three entities are the things about which the organization currently collects and maintains data for the rental activity. An ERD with these three entities appears in Figure 4.3 using the symbols from Figure 4.2.

FIGURE 4.3
Entity
Relationship
Diagram for
Video Rental



A second step is to identify relationships. The basic activities in this situation are: a customer makes rentals and each rental involves one or more videos. In other words, a relationship exists between the Customer and Rental entities and a second relationship exists between Rental and Video entities. In the actual organization function, a customer ends up with a video only through the rental event. The ER model always should reflect the real situation. A relationship that links Customer to Video is incorrect because no such link exists in the actual rental activity.

Relationship lines specify not only the links between entities but also the **maximum cardinalities** for the link. Normally, the only two maximum cardinalities considered are (1) one and (2) many, which is more than one, although notation does exist to specify other maximums. The ER model specifies the maximum cardinality for both ends of a relationship using a straight line symbol for a maximum of one, and a crow's foot or three-line symbol for a maximum of more than one as shown in Figure 4.2. The one or the many symbol that connects an entity to a relationship specifies the maximum number of instances of that entity that can interact with one instance of the entity at the other end of the relationship. With **one-to-many relationships**, the analyst must take special care to get the one and many symbols next to the correct entity. With **one-to-one** and **many-to-many relationships**, this problem disappears.

The maximum cardinalities for relationships involved in this organizational situation as shown on Figure 4.3 are as follows:

- A customer may enter into *many rentals*, but each rental is for *one customer*, that is, a one-to-many relationship between CUSTOMER and RENTAL. As noted, the end of the relationship symbol with a single line is the “one” end; the other end with the crows foot or three lines is the “many” end. The analyst graphs this relationship on the ERD by placing the many symbol next to the Rental entity, to show a maximum of many rentals per customer, and the one symbol next to the Customer entity, to show a maximum of one customer per rental.
- A rental may involve a maximum of many videos. But at different points over time, a video also may be part of many rentals, for example, a specific video can be rented many times. The result is a many-to-many relationship between RENTAL and VIDEO. The relationship symbol that joins RENTAL and VIDEO has a crow's foot at both ends, indicating a many-to-many relationship.

The last step is to identify attributes. Examining the forms in the Figure 4.1 exhibits and reading the narrative suggests the following attributes for the entities:

- CUSTOMER—Name, Member-No, Address, Tel-No, Credit-Card-No, Expire-Date. Since Member-No has a unique value for every instance, Member-No can serve as a primary key for the entity CUSTOMER. Address is a **composite attribute**, an attribute that consists of several other component attributes, in this case, Street, City, State, and Zip.
- RENTAL—Rental-No, Date, Employee-No, Pay-Type. Rental-No is the primary key for RENTAL.
- VIDEO—Video-No, Title, Date-Acquired, Rental-Charge/Day, Vendor. Video-No is the primary key for VIDEO.

In Figure 4.3, a line connects each oval attribute symbol to its entity. Every symbol, attribute, entity, and relationship, has a name to identify it. The names for attributes that serve as primary keys of entities are underlined. With a composite attribute, additional lines connect the composite attribute to its component attributes.

Several attributes from the forms and narrative, Due-Date, Cost, Return-Date, and Overdue-Charge, do not seem to belong to any entity. They are in fact attributes of the many-to-many relationship between RENTAL and VIDEO. Only many-to-many relationships may have attributes. When one instance of a Rental is linked to one instance of a Video, then Due-Date, Cost, Return-Date, and Overdue-Charge are attributes of the pair of instances linked by the relationship. Lines connect the attributes of the many-to-many relationship to the diamond symbol for the relationship.

Two data items in Figure 4.1, Exhibit 2, Tax and Total, do not appear above as attributes. These items are calculated during the rental process. If desired, Tax and Total could be included in the data model as **derived attributes** of rental, that is, attributes whose values are calculated from the data in other attributes. Often derived attributes are not included in the data model because a process can calculate the values as needed.

Note that Figure 4.3 presents a different-looking data model than the forms shown in Figure 4.1. Both describe the same data. The forms describe the data in terms of a specific implementation or application, that is, the forms are application dependent. The forms also contain duplicated data. The ERD describes the underlying data structure associated with the entities involved in the organizational function of renting a video. The ERD model is independent of any specific application or implementation. The ERD model also eliminates some data duplication in that each attribute shows up only once in the ER diagram.

Data models in common with other models can exist at these three technology levels:

1. **Conceptual**—A **conceptual data model (CDM)** presents the data in only an organizational context with no reference to technologies or mechanisms for physical implementation.
2. **Logical**—A **logical data model** operates within a class of technologies, for example, flat files or relational tables. Relational models are discussed in the next section.
3. **Physical**—A **physical data model** follows the constraints of one specific technology, for example, the use of an MS Access database or an Oracle Server database.

The ERD models presented in this chapter represent conceptual data models; the models neither assume a logical structure for the data nor a physical implementation. The data model in Figure 4.1 and the associated narrative specify both a logical structure (flat files) and a physical implementation (paper forms in files). A list of the contents of the forms from Figure 4.1 could be a conceptual data model.

ERD Rules

For ERDs to realize their value as an analysis and communication tool, analysts should follow a common set of rules or procedures to construct them. Ideally, or if the rules were complete, different analysts looking at the same system should create identical ERDs. In practice, the rules for ERDs facilitate a standard approach, but each analyst still must make a number of decisions about how to represent the system in ERD format.

The rules attempt to achieve the following goals:

- Assure that the model accurately and completely represents the relevant content of the system. ERDs do not claim to model graphically all the data in an organization. However, ERDs should accurately model all of the data related to the organizational function under study that are of interest to the client.
- Within the limits of the rules, every analyst should model the data for the same function with the same or a similar representation.

Basic Rules and Guidelines

The basic rules and guidelines provide the underlying common structure for the ERDs. A number of additional rules discussed later deal with specific situations. An examination of the ERD in Figure 4.3 will show that the ERD follows the rules below.

- *A thing is an entity only if the organization wishes to keep data about the thing.* In the GB Video example, Customer, Rental, and Video are entities because the client wishes to collect and maintain data about them. Most of the time, forms and reports are not entities. The invoice form in Exhibit 2 of Figure 4.1 is not an entity because the organization does not wish to keep information about invoice forms. Forms and reports tend to reflect a specific application, not the underlying data. Forms and reports contain data from the underlying entities. The invoice form contains data from the Customer, Rental, and Video entities. In an organization with a high level of security, classified reports could be entities if the organization wishes to keep information on the people who see or have access to each report.
- *An entity must contain more than one instance.* In the GB Video example, GB Video probably is a “thing” about which the organization wishes to keep data, but GB Video is not an Entity because it consists of only one instance. Customer, Rental, and Video contain multiple instances and are entities.
- *Every entity must have an attribute or a set of attributes that serve as a primary key for the entity and the primary key must be unique.* In other words, a specific value for a key can appear only once in all instances of the entity. Some entities will contain more than one **candidate** for a **primary key**. For example, a Customer entity may contain a unique customer number and also a Social Security number. Both are unique candidate keys. The analyst chooses which one to use.
- *A primary key can be a set of attributes or a composite attribute.* For example, customer last name, first name, and telephone number might serve as a

composite primary key for the Customer entity class. The analyst should check carefully to ascertain that the composite key is unique. For example, the composite of two primary keys of the entities related to an associative entity may not always be unique. The cautious analyst uses an arbitrary sequential set of characters, for example, member number, in place of a composite attribute as the primary key to assure uniqueness.

- *An entity has two or more attributes.* If an entity has only one attribute, then that attribute must be the primary key. Other than the key, the entity contains no data and thus does not meet the definition of an entity—things about which the organization wishes to collect data. In the GB Video example, Clerk or Employee is not an entity in the video rental function because the only data needed on clerks are the Employee ID numbers. Employee probably is an entity in the personnel or payroll part of the organization. Exceptions to this rule may exist, but such exceptions are rare.
- *A many-to-many relationship can have one or more attributes.* The attributes of a relationship are attributes that exist only when an instance of one entity is linked to an instance of one or more other entities by the many-to-many relationship.
- *A relationship must represent a situation that actually exists or the client wants to exist in the organization.* The relationship between Customer and Rental actually exists in GB Video in that customers do make rentals. Placing a relationship on the GB Video ERD between Customer and Video is an error; no direct relationship exists between Customer and Video in the GB rental activity as described. The only way that a customer interacts with a video is through a rental event (or perhaps inappropriately through a shoplifting event). If a social organization wants to keep data about the videos owned by each of its members, then a relationship will exist between Member and Video.

Naming Rules

Naming rules define how names and/or labels are assigned to entities, relationships, and attributes. Labels and names add clarity and resolve ambiguity. All the components in Figure 4.3 have unique names that follow the rules provided below. At a later stage, the analyst may wish to generate additional information about each of the components in a metadata section or table. The unique labels or names for each component allow the analyst to tie the additional information to each specific component in the ERD. Unique labels or names also allow the programmer to write documentation that ties code modules to specific components on the ERD.

The rules below present some commonly used conventions for labels and names. In practice, organizations often establish their own detailed rules for labels and names. Many times, analysts may do their own thing. Just about every naming convention imaginable will appear if one examines enough ERDs.

- *Every component on an ERD must have a unique name and/or label.* Some analysts use only names. Some ERD drawing tools and analysts may assign sequential, unique ID characters to every component or to some components as a label, for example, E1, E2, . . . En for entities; R1, R2, . . . Rn for relationships; and so on.

- An Entity name consists of a singular noun in all capital letters that describes the instances in the entity class, for example, CUSTOMER. People, including the authors, who dislike using all capital letters sometimes use upper- and lower-case for entities, for example, Customer.
- A Relationship name consists of a verb or a phrase in upper- and lowercase letters, for example, “Make” or “Requestor of.”
- Attribute names are nouns or a string of nouns, adjectives, and other characters connected by hyphens or underscore characters using upper- and lowercase, for example, Employee_ID or Employee-ID or Address1.
- When an attribute serves as the primary key for an entity, the attribute name is underlined, for example, Member-No.

A number of other rules and conventions, exist for ERDs. These conventions will be discussed and illustrated later in the chapter.

Additional Constructs for ERDs

ERDs contain or provide for many features and constructs in addition to those discussed thus far. This chapter covers only the major ones including multivalued attributes, weak entities, associative entities, relationship degrees, minimum cardinality, and supertypes/subtypes.

Multivalued Attributes

A **multivalued attribute** can have several values for a single instance of an entity. For example, the client might wish to allow family members of customers to rent videotapes or DVDs using the customer’s member number. The attributes, Family-first-name and Family-last-name, may consist of the first and last names of one, two, or a number of family members. The name attributes are thereby multivalued. Taken together, the attributes form a *repeating group*, in this case, a list of related first and last names. Figure 4.4 is a partial ERD showing one possible representation for this multivalued repeating group of attributes using the multivalued attribute symbol. Some technologies, for example, COBOL, will support multivalued attributes, but relational database technologies, for example, MS Access or Oracle Server, will not.

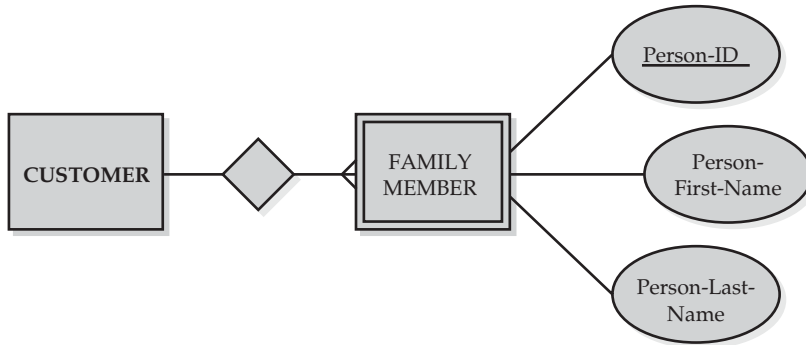
Weak Entities

A **weak entity** is an entity that can exist only in conjunction with a regular or strong entity. In Figure 4.5, a weak entity replaces the multivalued attribute in Figure 4.4 for names of family members.

FIGURE 4.4
Multivalued
Attributes



FIGURE 4.5
Weak Entity



If a customer ends his or her relationship with GB Video, the family members' relationships also end. Family members exist only in conjunction with a specific customer. The primary key of a weak entity is the composite of the primary key of the regular entity and the local key for the weak entity such as Member-No. plus Person-ID in Figure 4.5. Person-ID probably is a sequential key, such as 1, 2, 3 . . . n for the family members for a customer. This key by itself is not unique: A number of customers may have 1, 2, 3, or more family members. Sometimes, the double box shown in Figure 4.5 or another special symbol is used for weak entities, but much of the time the standard entity symbol is used.

Associative Entities

While many-to-many relationships accurately can describe some organizational activities in a data model, an alternate representation called an associative entity can add specificity and avoids implementation problems with some logical and physical data models. An **associative entity** links an instance of one entity in the many-to-many relationship to the related instances, if any exist, in the other entity. The associative entity replaces the many-to-many relationship with two one-to-many relationships linking the associative entity to the two other entities as shown in Figure 4.6. The diagram shows the many-to-many relationship discussed previously between Rental and Video converted into two one-to-many relationships with an associative entity called Rental/Video.

The new relationships mean

- A Rental instance may contain *many* Rental/Video instances, but each Rental Video instance is linked to *one* Rental instance.
- A Video instance may be held by *many* Rental/Video instances, but each Rental/Video instance is linked to *one* Video instance.

FIGURE 4.6
Associative Entity



In other words, the two one-to-many relationships specify that *each instance in the entity Rental/Video links exactly one instance of Rental to exactly one instance of Video*. If a rental involves three videos, then Rental/Video will contain three instances that link to the same instance in Rental, and each of these three Rental/Video instances will link to a different instance of Video.

Figure 4.6 shows the associative entity linking instances from two entities; however, an associative entity can link instances from three or more entities by adding more one-to-many relationships connecting the other entities to the associative entity.

One naming convention for an associative entity is the composite of the linked entity names, in this case, Rental/Video. However, the analyst may give the associative entity any name that seems descriptive, for example, Line, Subrental, and so forth. The diamond in the relationship becomes the associative entity and the “many” sides of the one-to-many relationships *always* connect to the associative entity. The diamond in the box is a standard symbol for an associative entity, but some case tools and analysts use the standard entity symbol for all entities. At the level of logical and physical data models, all entities—regular, weak, and associative—behave the same. To paraphrase Gertrude Stein, An entity is an entity is an entity.

The attributes of the relationship, if any, become attributes of the associative entity. The primary key of the associative entity may be (1) the composite of the primary keys for Video and Rental if unique; or (2) a new arbitrary unique ID over all instances of Rental/Video; or (3) a composite of the primary key for Rental plus a unique key for each of the instances that link to one Rental instance, for example, 1, 2, . . . n; or (4) any other unique key. In Figure 4.6, the composite of the Rental-No and Video-No attributes is unique and can serve as the primary key for Rental/Video. When the key is the composite of the primary keys of the linked entities and the special associative entity symbol is used, the identifying attributes are assumed and need not appear on the diagram.

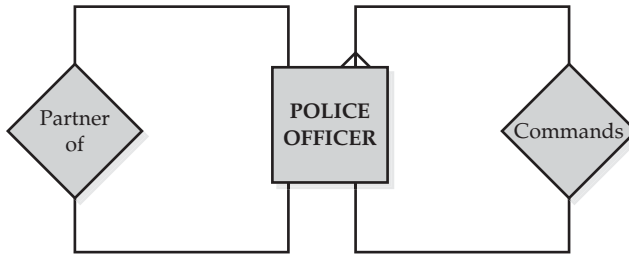
Degree of a Relationship

The examples used thus far show a relationship linking two entities—a **binary relationship**. A relationship also can link an entity to itself—a **unary relationship**. Or a relationship can link three entities—a **ternary relationship**. The number of entity classes that participate in a relationship defines the degree of the relationship.

Two examples of unary relationships are shown in Figure 4.7. The examples show the entity Police Officer. The entity contains data about the officer’s, for example, Name, Address, Rank, and so forth. An officer may have a partner with whom he or she works. An officer may have one partner, or an officer may be a partner of one other officer. A unary one-to-one relationship describes this organizational situation.

A second unary relationship for the same entity shows the command situation. An officer may be the commander of a number of other officers and an officer has one commander, which is a one-to-many unary relationship. This example also shows that an entity may be involved in a number of relationships including

FIGURE 4.7
Unary
Relationships



multiple unary relationships. An entity also may participate in a unary many-to-many relationship. Unary many-to-many relationships are difficult to understand. Unary many-to-many relationships are discussed in the next section on logical data models, where they are easier (but not easy) to understand.

In a ternary relationship, three entities interact in a manner defined only by linked instances of all three entities. For example, the recital schedule during a music festival might be described in terms of data about three entities: Artist, Work, and Hall. A specific performance brings the artist, work, and hall together at a given time period. Performances may take place with many works in many halls and by many artists. The ternary relationship for this situation is illustrated in Figure 4.8.

The attribute Time-Period is an attribute only of the ternary relationship; it is not an attribute of an entity or of a binary relationship between any of the pairs of entities, that is, at one specific Time-Period, one instance from each of the three entities link together.

When the ternary relationship is converted into an associative entity, the resulting ERD is shown in Figure 4.9. In a number of cases (as in this one), the associative entity is more than an abstract concept; a Performance is a physical “thing” in the organizational situation. A specific artist, work, and hall may appear together in a number of performances, that is, the composite of the primary keys for Artist, Work, and Hall is not unique. In this situation, the attribute, Time-period, is also needed to create a composite primary key for Performance. A cautious analyst ignores composite keys and gives the Performance entity a unique key.

FIGURE 4.8
Ternary
Relationships

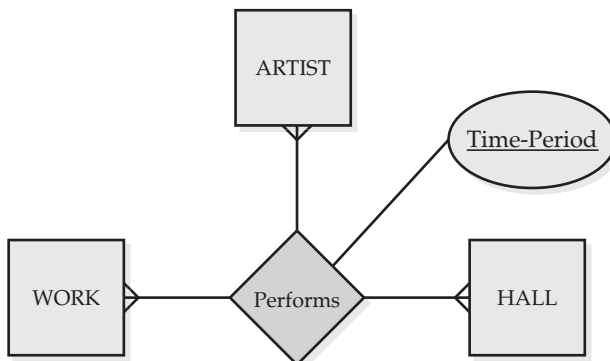
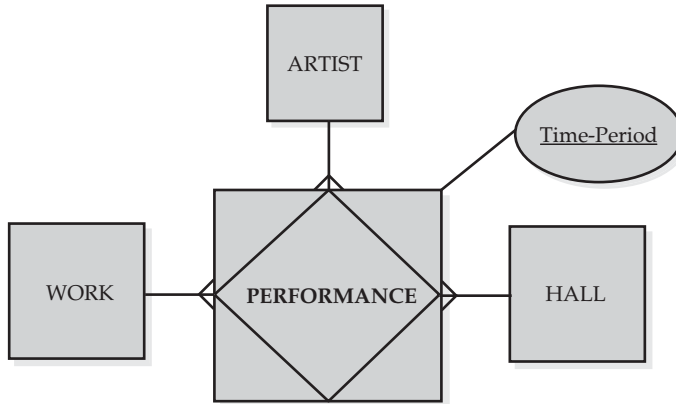


FIGURE 4.9
 Associative
 Entity
 Converted
 from Ternary
 Relationship



Higher degree relationships with four or more entities can exist in some situations. Most people find the use of an associative entity to diagram ternary or higher degree relationships clearer than the relationship by itself.

Minimum Cardinality

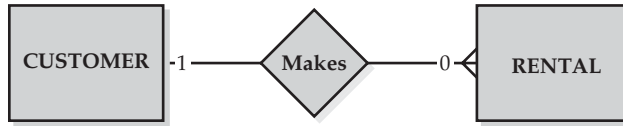
Just as maximum cardinality specifies the maximum number of other instances that an instance can relate to, **minimum cardinality**, sometimes called *optionality*, specifies whether an instance from an entity at one end of a relationship must interact with any instance from the entity at the other end of the relationship. Can an instance relate to zero other instances? Minimum cardinality normally is either one or zero. For example, must a Customer instance relate to one or more Rental instances; or in other terms, can a person be a customer and have never rented a video?

This type of requirement is an example of an organizational rule because it represents a constraint that is determined by the way the organization chooses to operate. If GB Video decides that customers can exist without making any rentals, then a Customer instance has a zero minimum cardinality or an **optional relationship** with respect to Rental instances; in other words, a Customer instance may relate to zero Rental instances. Alternative, GB might decide to classify people as customers only if they have rented a video, so there is a minimum cardinality of one rental per customer.

At GB Video and at most video stores, a rental must relate to a specific customer, that is, every Rental instance has a **mandatory relationship** with exactly one Customer instance, or a minimum cardinality of one customer per rental and a maximum cardinality of one customer per rental. If a rental is made with no customer involved, the store may encounter great difficulty getting the video back (who has it?).

Every relationship on an ERD has two minimum cardinality indicators—one for each end of the relationship. An optional relationship, minimum cardinality of zero, is shown in this chapter and text by placing a 0 on or near the relationship line next to the entity; and a mandatory relationship, minimum cardinality of one, by placing a 1 on or near the relationship line next to the entity. Many conventions for showing optionality exist. For example, Oracle design and developer tools use two segment relationship lines with a dotted line segment for optionality and a solid

FIGURE 4.10
Cardinality
and
Optionality
Symbols



line for a mandatory relationship in the reverse of the position of the lines from where one might expect. A dotted line at one end of a relationship indicates optionality at the other end. Careful study may be required to understand the meaning and operation of the minimum cardinality notation used on a specific diagram.

This relationship for Customer and Rental is illustrated in Figure 4.10. The combination of cardinality and optionality symbols in the figure show that a customer can make zero, one, or more than one rentals (minimum of zero and maximum of many), but a rental must be for exactly one customer (minimum and maximum of one).

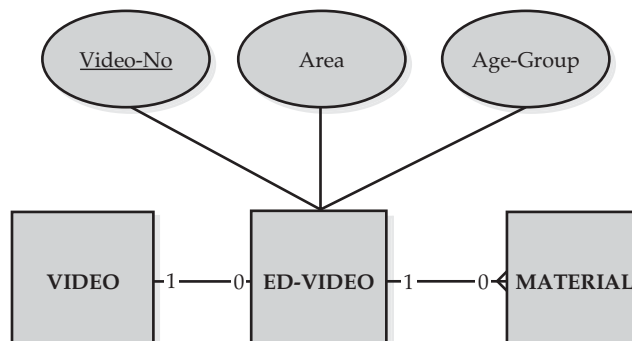
Supertypes and Subtypes

Some entities appear to contain subgroups within them. For example, GB Video might rent two types of videos—regular videos and educational videos. The company wishes to maintain some additional data about educational videos—age group, for example, 2–4, 5–8, 9–12, 12–16, and area, for example, history, literature, science. In addition, an educational video can have one or more items of additional teaching materials that supplement the content of the video; in other words, a relationship exists between Video and Material.

The analyst has two ways to deal with this organizational situation. The analyst can include Age-Group and Area as attributes of Video and set up an optional one-to-many relationship between Video and Material. A regular video will link to zero instances in Material, but an educational video may link to one or more instances of Material.

Alternatively, the analyst can set up supertype/subtype entities. A **supertype** is an entity that has subtypes and the supertype contains the common data that apply to all variations of the entity. A **subtype** contains only the attributes that apply to the subtype. A subtype has a one-to-one relationship with the supertype and may enter into relationships with other entities. A supertype/subtype representation for Video (a supertype), the subtype named Ed-Video and a regular entity, Material, is shown in Figure 4.11.

FIGURE 4.11
Supertype/
Subtype
Representations



The primary key of Ed-Video is the same as the primary key of Video. The minimum and maximum cardinalities of the relationship state that an instance in Video may link (optional or minimum cardinality of 0) to one instance in Ed-Video, but every instance in Ed-Video must link to exactly one instance of Video.

Sometimes the analyst ends up with several entities that contain much of the same data, for example, the entities Motorcycle, Car, and Truck might exist in a vehicle registration area of an organization. When entities contain a lot of the same data, the entities probably are subtypes of a supertype. For example, Motorcycle, Car, and Truck might be subtypes of the supertype, Vehicle.

Rules for supertype/subtype entity classes include the following:

- *A supertype entity has one or more subtype entities.*
- *The supertype and subtype entities are linked by one-to-one relationships.*
- *Each subtype entity may participate in additional relationships with other entities.*
- *The attributes common to all the subtypes are the attributes of the supertype.*
- *Each subtype may have additional attributes.*
- *Supertypes can participate in two types of specialization:*
 - *With total specialization, every instance of the supertype must link to one instance in one of the subtypes.*
 - *With partial specialization, an instance in the supertype may link to zero instances in all of the subtypes. In the GB Video example, some instances of regular videos link to zero instances in the subtype.*
- *Instances of a supertype can link to multiple instances of subtypes as follows:*
 - *Disjoint—A supertype instance can link to an instance of only one subtype.*
 - *Overlap—A supertype instance can link to one instance in each of several subtypes.*

Entity-relationship models provide special notation to represent specialization and disjoint/overlap relationships graphically (see, for example, Hoffer, 2005b).

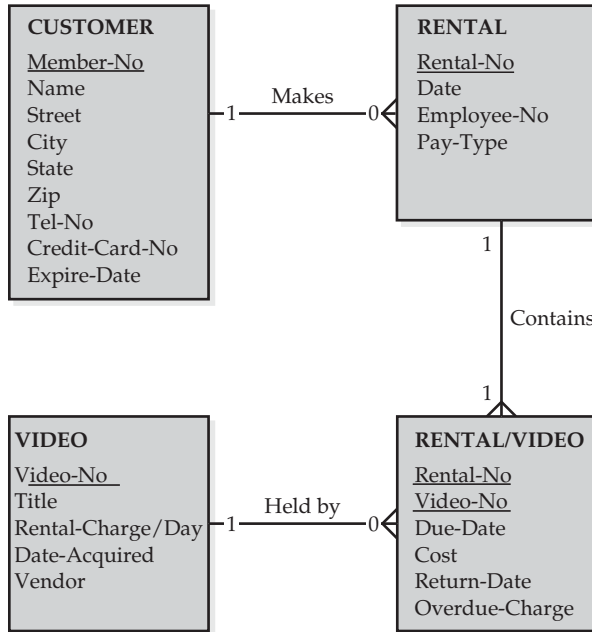
Simplified, Reduced-Form ERDs

The ERD in Figure 4.3 correctly describes the organizational activity for renting a video, uses standard symbols, and follows the rules. For many people, especially busy analysts, the ERD seems too busy and unwieldy. Regular users of ER models and system development tool vendors tend toward more parsimonious representations. One such representation is the **simplified, reduced-form ERD (SERD)**.

Rules for simplified, reduced form ERDs or SERDs include

- *Omit the relationship diamonds but keep the relationship names.*
- *Eliminate the ovals for attributes; list attributes in the entity box and place the entity name at the top of the box in bold print.*
- *Replace composite attributes with the component attributes.*
- *Replace many-to-many relationships by an associative entity using the standard entity symbol.*

FIGURE 4.12
Simplified,
Reduced-
Form ERD for
Video Rental



- Replace multivalued attributes with an entity using the standard entity symbol for both regular and weak entities.

Figure 4.12 illustrates a simplified, reduced form ERD for the GB Video rental function. The new diagram contains the same information as Figure 4.3 except the composite attribute Address has disappeared: The component attributes of Address—Street, City, State, and Zip—appear in the diagram.

The diagram also shows both maximum and minimum cardinalities. Every rental must be made by exactly one customer (minimum of one and a maximum of one), but a customer may have zero, one, or many rentals (a minimum of zero and a maximum of many). In other words, the customer can be a member and not have rented a video. Every Rental instance must contain at least one Rental/Video instance (minimum of one and maximum of many) and each Rental/Video instance must be contained by exactly one Rental instance (minimum and maximum of one). Every Rental/Video instance must link to exactly one Video instance (minimum and maximum of one), but a Video instance may link to zero Rental/Video instances (minimum of zero and maximum of many). The store may have some videos in stock that have never been rented.

CONCEPTUAL DATA MODELS

Conceptual data models (CDMs) provide the best view of the underlying data structure associated with an activity in an organizational context. The conceptual data model for an existing situation specifies the data that the organization

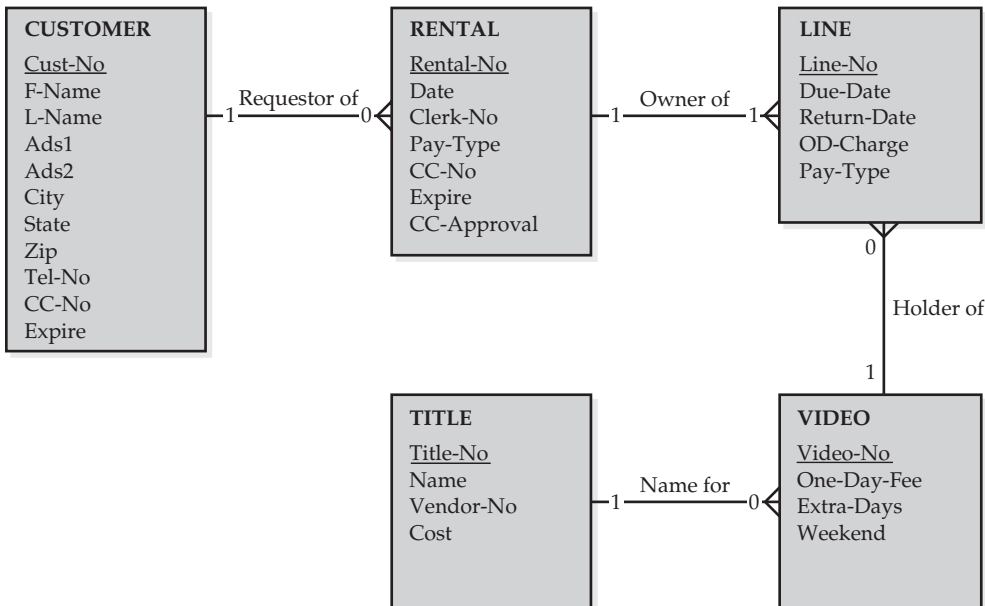
currently keeps about the activity. The conceptual data model for a proposed situation specifies the data that the client wishes to keep about the organization’s function or activity under study.

As already noted, conceptual data models are independent of the technologies selected to implement the data structure. Some people suggest that conceptual data models also are application independent. While conceptual data models help the analyst to devise a data structure that can support multiple applications, CDMs are not completely application independent. The analyst and client must think about the possible applications to make sensible decisions on the entities and attributes to include in the model.

Conceptual models should strive to eliminate duplication of data. The model shown in Figure 4.12 for GB Video eliminates much of the duplication found in the form- and file-based model of the current system as described by the narrative and Figure 4.1. But, as previously discussed, some duplication still exists. Every instance of Video contains attributes for Title, Cost, and Vendor. But each title is produced by one and only one vendor and has a single cost. Instead of duplicating the vendor and cost data in each instance with the same title in the Video entity, the analyst can create an entity called Title to hold this data once for each title. The relationship is one-to-many, that is, each instance in Title may relate to many instances in Video, but an instance in Video relates to one and only one instance of Title.

Figure 4.13 shows a conceptual data model for GB Video in simplified, reduced form. The ERD, as already shown, contains an entity for Title. This diagram appears in Chapter 8 and is based on the narrative model for a proposed GB

FIGURE 4.13 Conceptual Data Model for GB Video



Video system. Figure 4.13 contains a number of minor changes from the diagram in Figure 4.12, including some additional attributes discussed in Chapter 8 and different attribute names and phrases to name the relationships. In this diagram, the associative entity is named Line. One of the advantages of ERDs is that these changes in format and nomenclature tend to have little impact on meaning. The meaning of the diagram remains clear to most people familiar with ERDs.

METADATA

Metadata are data about data. More precisely, metadata describe the meaning, identification, and form of the components of a model. Detailed metadata about a data model can be extensive and are usually maintained in some form of electronic repository. Some database management systems, for example, Microsoft Access, incorporate that repository in the interface of the system itself, while most larger database engines maintain metadata in a separate location. The appropriate level of detail for metadata varies with the complexity of the project and the standards of the IT organization.

Conceptual-level metadata focuses on the components of the data model while physical-level metadata deals with how to implement the components in a specific physical environment. At the conceptual level, many physical parameters, such as formatting and storage space requirements, are not of immediate concern. However, recording such information when it is encountered or discussed can help the physical system developers at a later time. Sample metadata for the ERD in Figure 4.13 appear in Table 4.1.

ENTERPRISE DATA MODELS

Enterprise data models (EDMs) use the entity-relationship model framework to provide an overview of the data structure associated with an organization. An EDM displays the major entities and relationships. Normally, an EDM gives a broad context for a proposed system or may cover the entire organization. Figure 4.14 shows a possible EDM for the GB Video rental/return activity. Two entities

FIGURE 4.14
Enterprise
Data Model
for GB Video
Rental/Return
Activity

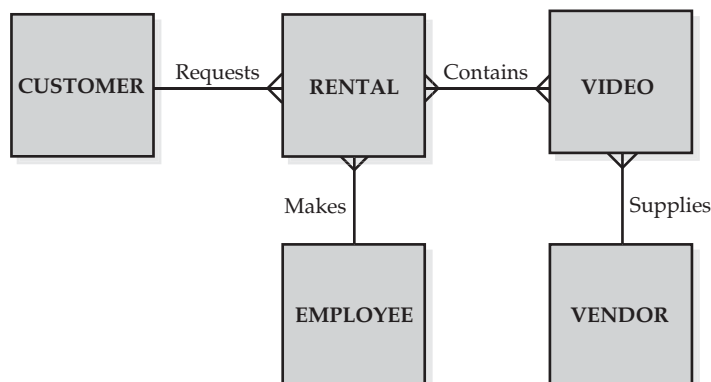


TABLE 4.1
Metadata for
the GB Video
Rental ERD

Entity Metadata. Entity metadata may contain the names of the entity classes in the model and a description of the data they hold. Additional design data might include file size, file location, and access authority.

Entity	Description
CUSTOMER	Contains all the available information about each customer who has made a transaction in the last year
LINE	Contains the information on each video associated with a rental transaction
RENTAL	Contains the information on each rental transaction
TITLE	Contains information on each distinct title of the videos
VIDEO	Contains information on each individual video

Attribute Metadata. Attribute metadata may contain the attribute name and a description of the attribute. Additional metadata might include data type (text, number, etc.), length, format, whether null values are allowed, etc.

CUSTOMER. Contains all the available information about each customer who has made a transaction in the last year.

Attribute	Description
Cust-No	A unique key assigned to each member
F-Name	First name and middle initial if any
L-Name	Last name
Ads1	Street or box address
Ads2	Apartment number or other as needed
City	Name of city
State	State id code
Zip	Zip code
Tel-No	Telephone number
CC-No	Credit card number
Expire	Expiration date on the credit card

RENTAL. Contains the header information on each rental transaction.

Attribute	Description
Rental-No	Unique key assigned to each rental
Date	Date of the rental
Clerk-No	Employee number of the clerk entering the rental
Pay-Type	Cash, check, or credit card
CC-No	Credit card number
Expire	Expiration date of the credit card
CC-Approval	Credit card approval code

LINE. Contains the information on each video associated with a rental transaction.

Attribute	Description
Line-No	Unique key assigned to each line
Due-Date	Date tape is to be returned
Return-Date	Actual return date
OD-Charge	Charge for days kept after due date if applies
Pay-Type	Method of payment for the overdue charge

VIDEO. Contains information on each individual video.

Attribute	Description
Video-No	A unique key assigned to each video
One-Day-Fee	First day rental fee
Extra-Days	Extra days rental fee
Weekend	Rental fee for Sat. and Sun.

TITLE. Contains information on each distinct title of the videos.

Attribute	Description
Title-No	Unique key for the title
Name	The name of the video, e.g., <i>Charlie's Angels</i>
Vendor-No	Key for the vendor who produced/sold the video
Cost	Purchase price for the video

Relationship Metadata. Relationship metadata may give the relationship name, describe the relationship, and for both of the entities in the relationship: states the entity name, explains in a sentence the meaning of the minimum and maximum cardinalities and shows the minimum and maximum cardinalities in parentheses. When the minimum cardinality is one (mandatory), the text uses “must” before the verb; when the minimum cardinality is zero (optional), the text uses “may” before the verb. The minimum and maximum cardinalities in the metadata must correspond exactly to the ones shown next to each corresponding entity in each relationship on the ERD, in this example, the ERD in Figure 4.13. The text sentence clarifies the cardinalities for someone who is not familiar with the notation or definitions.

Relationship Name	Description	Entity1 with (min, max) cardinality	Entity2 with (min, max) cardinality
Requestor of	Links each customer to rentals made by the customer	CUSTOMER—a rental <i>must</i> be for one customer (1, 1)	RENTAL—a customer <i>may</i> make many rentals (0, many)
Owner of	Links each rental to the associative entity	RENTAL—a line <i>must</i> belong to one rental (1, 1)	LINE—a rental <i>must</i> contain one or many lines (1, many)
Holder of	Links the associative entity to a specific video	LINE—a video <i>may</i> be held by many lines (0, many)	VIDEO—a line <i>must</i> hold one video (1, 1)
Name for	Links a title to the videos that use the title	VIDEO—a title <i>may</i> be the name for many videos (0, many)	TITLE—a video <i>must</i> be named by one title (1, 1)

appear that probably will not appear in the proposed system, Employee and Vendor. Employee is relevant for the EDM because employees rent tapes or DVDs to customers and the employee number appears as an attribute of Rental. GB Video purchases the videos from Vendors and the vendor identification appears in the Video entity. In other words, both Employee and Vendor are part of the broad data structure for the organizational area. The analyst could and in many circumstances should expand the GB Video EDM to cover the entire organization.

The following rules apply to EDMs:

- *All of the strong or regular entities that play a role in the entire organization or the organizational area under study are included. Weak, multivalued attribute and reference entities are omitted.* In the GB Video example, Title, a reference data entity, does not appear on the EDM.
- *Relationships are shown with maximum cardinalities but without minimum cardinalities. Verbs or phrases to describe the relationship are included but relationship diamonds are omitted.*
- *Many-to-many relationships are acceptable.* The GB Video example shows a many-to-many relationship between Rental and Video.
- *Attributes of the entities are not shown.*
- *Associative entities are not included unless they represent an important “thing” in the organization.* The associative entity, Line, in GB Video does not represent an important “thing”; it is a part of the rental activity. In the music festival example, the associative entity Performance that replaces the ternary relationship is an important “thing” in the organization and probably should appear on an EDM.

LOGICAL DATA MODELS

Logical data models translate conceptual data models into a specific data storage structure. Many information systems from 1950 to 1980 used a logical structure of sequential or “flat” files stored physically on magnetic tape. In a flat file structure, the instances of data, often ordered by a primary key, are stored one after another in the file. The most widely used programming language for organizational data processing in the 1960s through the 1990s, COBOL, was built around the flat file logical data structure. Magnetic tape files were replaced by files on magnetic disk drives with random access, but often the flat file logical structure remained. The availability of disk storage led to increasing use of hierarchical and network logical structures, for example, the Information Management System (IMS) and the Integrated Data Management System (IDMS).

The Relational Model

Today, the most common logical model for data storage is the **relational model**. E. F. Codd, in 1970, set forth the theory for the relational model, a formal mathematical structure that allowed for mathematical proof of fundamental data operations. A large number of physical database implementations use the relational model. Many, if not most, new applications use relational databases.

TABLE 4.2
ER and
Relational
Model
Correspon-
dence

ER Model Terms and Concepts	Relational Model Terms and Concepts
Entity (regular, weak, or associative)	Table or relation
Single-valued attribute	Column or attribute
Multivalued attribute	(Not allowed)
Instance	Row or tuple
Primary key or primary identifier	Primary key
One-to-one or one-to-many relationship	Foreign key
Many-to-many relationship	(Not allowed)

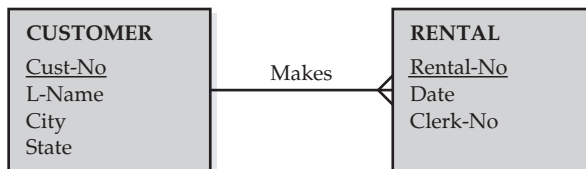
Basic Concepts

The relational model carries over the concepts of entity relationship models but adds additional structure. Table 4.2 shows the correspondence between the relational model and the entity relationship model.

In the relational model, data are stored in two-dimensional tables containing rows and columns. Each **table** corresponds to or contains the data for an entity. Each column in a table represents an attribute, and columns have names that correspond to attribute names. Rows correspond to entity instances in that each row holds the data for one instance. One of the columns in each table holds the data for the **unique key** for the row; in other words, one column corresponds to the primary key for an entity, called a primary key in the relational model. A one-to-many relationship between two entities is implemented in the relational model by inserting a **foreign key** column in the table that corresponds to the entity on the many side of the relationship. The data in the foreign key column for each row link that row to the single row in the table on the one side of the relationship with a primary key value that matches the foreign key value.

Part of an ER diagram for GB Video appears in Figure 4.15 with the two entities Customer and Rental and several attributes for each entity. The figure also contains the corresponding relational tables with sample data. As noted, the Customer and Rental tables in Figure 4.15 correspond to the Customer and Rental

FIGURE 4.15
ER Diagram
with Corre-
sponding
Relational
Tables



CUSTOMER			
Cust-No	L-Name	City	State
23278	Clinton	Little Rock	AR
10995	Dole	Wichita	KS
22671	Kerry	Boston	MA
00987	Bush	Crawford	TX

RENTAL			
Rental-No	Date	Clerk-No	Cust-No
1176	01022006	11	22671
2235	02032006	07	00987
4450	11212006	07	22671
0067	01022006	09	10995
3309	05102006	11	10995
2621	03302006	08	22671

entities. Each entity attribute becomes a column heading in the corresponding table. Each row contains the data for one instance of the entity. For example, the first row in the Customer table contains the data for a customer who lives in Little Rock, Arkansas, and whose last name is Clinton.

The Customer entity has a one-to-many relationship to the Rental entity, so that a customer can make one or more rentals, but each rental is for one customer. To express this relationship in the tables, the Rental table contains a new column for a foreign key called Cust-No that is not an attribute in the Rental entity. The foreign key expresses the relationship between the Customer and Rental tables. The foreign key value for Cust-No links each row in the Rental table to the primary key value for the corresponding row in the Customer table. For example, customer Kerry (CUSTOMER:Cust-No=22671) made three rentals with rental numbers of 1176, 4450, and 2621 as indicated by the foreign key value of RENTAL:Cust-No="22671" for those rows in the Rental table.

Rules for Relational Models

As noted, relational models add additional structure to the ER model. The structure comes from the table concept and from such rules as those below.

- *Every table name and the full name of every column must be unique.* The full name for a column is the table-name plus the local column-name. Two or more tables may have the same local column name, for example, Cust-No, but the full names, for example, CUSTOMER:Cust-No and RENTAL:Cust-No, must be unique.
- *A column must have a single value for each row.* Multivalued attributes are not allowed in relational tables.
- *The meaning of a column is determined only by the name.* The location or order of columns has no significance. For example, the column names in the Rental table may appear in any order.
- *A row is defined only by the content of the information in the row.* The order in which the rows are stored has no significance.
- *The content of each row must be unique.* Because primary keys must contain unique values, the content of a row with a primary key column or attribute always will be unique.
- *The data type and format for data in each column must be the same across all rows.* For example, Clerk-No in the Rental table cannot be defined as an integer for some employees and a text description for others (required by most physical implementations).

The primary key value serves as a key for each row. Primary keys can consist of one or more data attributes but more commonly consist of a set of arbitrary unique values generated specifically to act as keys. For example, sequential values of integers 1, 2, 3, . . . n, can serve as primary keys. Desirable primary keys have the following properties:

- Constant values over the life of the database. Some keys and some smart keys (the keys in which part or all of the key has meaning) tend to change over

time. For example, a smart key employee number might start with an IT (e.g., IT25, IT26, etc.) to identify employees in the IT department. If the employee changes departments, either the key must change or the key no longer conveys the correct meaning. Other smart keys, for example, the VIN number of an automobile that is set at the time the vehicle is manufactured, never change.

- Not composites of many attributes. Large composite keys may lead to unanticipated duplicate values and can slow down database operations.
- Uniquely representable. Floating point numbers make poor keys because of possible rounding errors. Keys normally use integer or alphanumeric data types (a physical model issue).

As noted, foreign keys are added to express the possible relationships between tables. In the relational model, foreign keys observe the following rules:

- *A foreign key column may have any unique full name.* The foreign key column local name often is the same as the local name for the primary key in the referenced table, but any unique name will work. For example, the analyst could use a foreign key local name of RentC# in place of Cust-No in the Rental table. The relationship between the foreign key and the primary key is defined not by the names but by text symbols or an arrow in a diagram at the logical level and by a data definition and/or operations command at the physical model level.
- *For one-to-many relationships, the foreign key always goes in the table on the many side of the relationship.* For example, the foreign key to link the Customer and Rental tables, goes in the Rental table. An attempt to place a foreign key of Rental-No in the Customer table will not work because the foreign key may have multiple values for the same row, that is, a customer can have more than one rental.
- *For binary one-to-one relationships, the foreign key may appear in either table.*
- *Relational tables do not allow for the implementation of many-to-many relationships.* A many-to-many relationship implies multivalued attributes, for example, multiple values for a foreign key, and thus violates the single value rule. The analyst must replace the many-to-many relationship with a table that corresponds to the associative entity in an ERD representation.
- *The data type of the foreign key must match the data type of the corresponding primary key (a physical model rule).*

The relational model does not provide a specific convention to represent minimum cardinalities or optionality. The analyst can specify mandatory relationships, for example, a rental must be for exactly one customer, by requiring that the foreign key value be non-null. A null value is an undefined value for a column of a row. In the GB Video rental example, the analyst probably would require non-null values for the foreign key, RENTAL:Cust-No. In other words, every rental must have a specific customer number entered that matches one of the primary key values in the Customer table. If null values are allowed, then the relationship is optional; a rental may be for a specific customer or may be for a null or undefined customer. GB Video management probably wants to prevent rentals that are associated with undefined and thus unknown customers.

Relational Schema

A **relational schema** is a graphical representation of the structure of the tables and the relationships between the tables for a relational model. A relational schema is the relational model equivalent of an ERD in the ER model. Representations for relational schema that are in common use include:

- *Column heading schema*—Uses a table name and the column heading row to represent the entire table.
- *Set notation schema*—Uses the table name followed by column names enclosed in parentheses to represent each table.
- *Box schema*—Uses a box with the table name and the column names inside to represent each table.

Figure 4.16 shows three different relational schema representations for the Customer and Rental tables in Figure 4.15. In the column heading and set notation schema, relationships are represented graphically by a **referential integrity** arrow from the foreign key column name to the primary key column name in the related table. The box schema shows referential integrity with a line that connects the foreign key and primary key. The symbol 1 near the line indicates a maximum cardinality of one and an * a maximum cardinality of many. The * or many symbol can only appear next to a foreign key. The arrow or line shows that a primary key/foreign key correspondence exists—that referential integrity exists.

In the different schema in Figure 4.16, some of the column names appear in different order; as noted in the rules, the order is of no significance. All the schema convey the same meaning but use different formats and notations. In each schema, the primary key for each table is underlined. Some notations mark the foreign keys especially in the box schema, for example, by adding an [fk] at

FIGURE 4.16 Alternative Representations for a Relational Schema

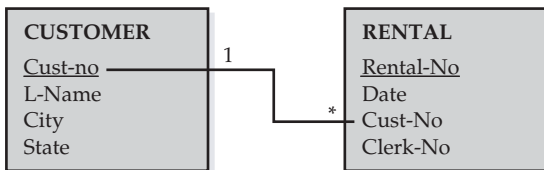
Column Heading Schema



Set Notation Schema

CUSTOMER(Cust-No, City, L-Name, State) RENTAL (Rental-No, Cust-No, Date, Clerk-No)

Box Schema



the end of the column name. When referential integrity arrows are used, the tails of the arrows clearly identify the foreign keys.

A column heading schema best displays the table structure. A set notation schema is the easiest and fastest to create in a word processing program. A box schema resembles ERDs. MS Access uses a box schema in the table relationship diagram.

Converting ERDs to Relational Schema

Different people hold differing views about the most efficient and effective way to generate a relational schema for a system. Some people, especially experienced analysts, can generate a relational schema correctly and rapidly without an ERD. Other analysts believe that starting with an ERD for the system and converting the ERD to a relational schema increases the likelihood of a complete and correct model.

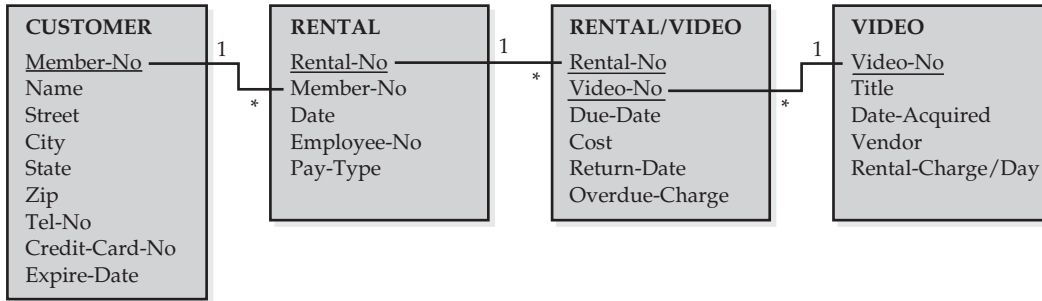
The steps in the process of converting a conceptual ERD to a relational schema are as follows:

1. Convert all the many-to-many relationships, if any, to associative entities.
2. Convert all multivalued attributes, if any, to entities.
3. Convert every entity (regular, weak, or associative) to a table with column headings that correspond to the attributes of the entity.
4. For *every* one-to-many relationship between two entities, add a foreign key to the table that corresponds to the entity on the many side of the relationship. A table that corresponds to an associative entity will have *two* or more foreign keys because it receives a foreign key for each of the one-to-many relationships that connect to the entity on the many side. The table that corresponds to an associative entity for a ternary relationship will have *three* foreign keys—one key for each degree of the relationship.
5. For a one-to-one relationship between two entities, add (or identify, if the primary key also is used as the foreign key) a foreign key to *either* one of the tables that correspond to the entities in the relationship.
6. Add a referential integrity arrow or line *from* each foreign key to the primary key of the entity on the other side of the relationship.
7. With unary relationships, one-to-one or one-to-many relationships that involve only one entity, the foreign key is added into the table for the entity and the referential integrity arrow goes from the foreign key to the primary key in the same table.

Applying these rules to the ERD in Figure 4.3 results in the relational schema (using box schema notation) in Figure 4.17.

The many-to-many relationship in Figure 4.3 is replaced with an associative entity, Rental/Video (rule 1). The four entities (three regular and the associative entity) become the Customer, Rental, Rental/Video, and Video tables with the same attributes as the entities (rule 3). The relational schema adds the foreign key, Member#, in the Rental table and two foreign keys, Rental-No and Video-No in the Rental/Video table (rule 4). The Rental/Video also uses the two

FIGURE 4.17 GB Video Rental/Return System Relational Schema



foreign keys, Rental-No plus Video-No, as a composite primary key. The Rental/Video table has two foreign keys because of the two one-to-many relationships that attach to the corresponding associative entity at the many side. The relationship symbols are replaced with referential integrity lines that go from the foreign keys to the primary keys (rule 6).

The relational schema in box format closely resembles the simplified, reduced form ERD shown in Figure 4.12. Simplified, reduced-form ERDs already incorporate rules 1 and 2. The analyst can start the conversion at rule 3.

Unary Relationships

Relational schema for unary relationships appear different than for binary relationships because only one table is involved. However, the rules for converting an ERD representation to a relational schema remain the same for all degrees of relationship. The diagram in Figure 4.18 from the ERD section of the chapter shows two unary relationships, a one-to-one and a one-to-many relationship. The corresponding relational schema is shown in the lower portion of Figure 4.18.

The relational schema follows directly from the ERD conversion rules. The Police Officer entity becomes a table with column names corresponding to the attribute names. Each of the unary relationships results in adding a foreign key to the table, Partner# for the partner of relationship and Commander# for the

FIGURE 4.18 Unary Relationships

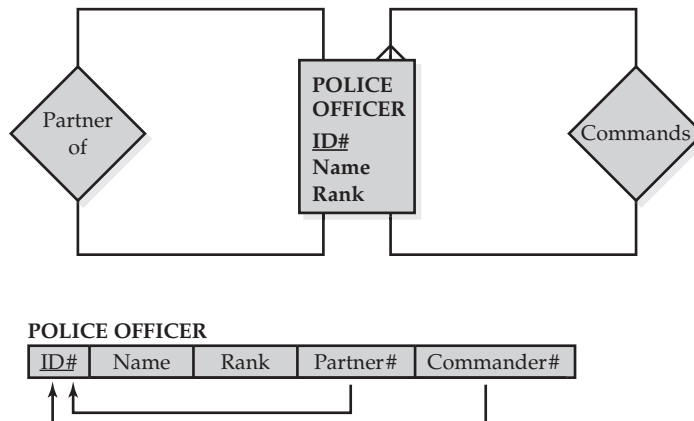
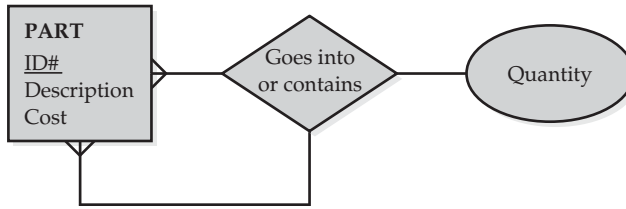


FIGURE 4.19
ERD for a
Unary Many-
to-Many
Relationship



commands relationship. The foreign keys contain the ID# of the officers who serve as partners and commanders.

The one-to-one and one-to-many relationships look the same in the column heading relational schema. Both of the relationships are expressed as foreign keys that reference the primary key of the table. Some information from the ERD, in this case both minimum and maximum cardinalities, disappears in the heading format and set notation graphic relational schema. The analyst can record this information in the metadata for the schema or use box schema to show maximum cardinalities.

Many-to-many unary relationships bring a higher degree of complexity. The best known such relationship is the bill of materials problem. A bill of materials looks at the relationships between parts used in a manufacturing or assembly operation. Some quantity of a part can go into an assembly of parts, and an assembly of parts can contain other parts including other assemblies. The ERD in Figure 4.19 is a mix of conventional and simplified notation.

The first step (rule 1) is to convert the many-to-many relationship into an associative entity, PART/PART, as shown in Figure 4.20 in simplified reduced form notation. Because the many-to-many relationship is unary, both of the new one-to-many relationships connect to the single entity, Part.

In Figure 4.20, a part can “go into” one or more other parts and also a part can contain one or more other parts. A part can range from a single component to an assembly of many components including other assemblies. The relational schema in Figure 4.21 comes from applying the ERD conversion rules.

FIGURE 4.20
Unary Many-
to-Many
Relationship
Converted to
an Associative
Entity

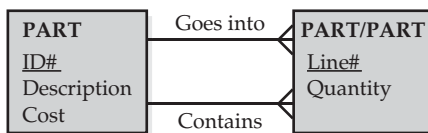
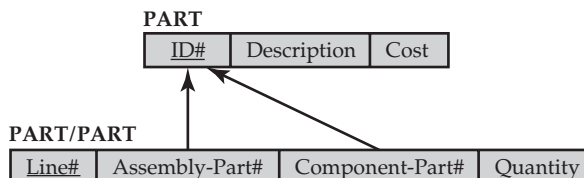


FIGURE 4.21
Relational
Schema
Applying ERD
Conversion
Rules



Assembly-Part# and Component-Part# are the two new foreign keys that express the two one-to-many relationships in the ERD. Line# is a new arbitrary, sequential primary key for the table, PART/PART. The composite of the two foreign keys is unique and could be used as the primary key. The attribute Quantity specifies the number of the component parts that go into each assembly part.

Normalization

Normalization addresses the improvement of a logical data design to avoid possible problems with data duplication and with the deletion and updating of data. The normalization process converts an un-normalized relation or table into two or more smaller, normalized tables. The normalization process consists of six steps that successively transform a relation into first, second, third, Boyce/Codd, fourth, and fifth normal forms.

First normal form requires the removal of any multivalued attributes from a relation. Because of the table structure the relational model will not accommodate multivalued attributes: A column in a single row can hold only one value. A relation must be in first normal form for physical implementation in a relational database.

Second normal form issues arise only when a table in first normal form has a composite primary key with at least one attribute that is not part of the key. If the value of one of the nonkey attributes depends on only part of the composite primary key—a condition known as **functional dependency**—the table is not in second normal form. The Rental/Video table (see Table 4.3) illustrates a second normal form violation.

The table is in first normal form. It has a composite key (Rental-No + Video-No) and has one or more nonkey attributes. Due-date depends on the full primary key, that is, the due date is for a specific video that is part of a specific rental. However, Title depends only on the Video-No because a specific video always has the same title regardless of the rental that it is in, and thus constitutes a second normal form violation. Creating two tables, as shown in Figure 4.22, corrects the problem shown in Table 4.3.

Both tables now observe second normal form.

Third normal form issues arise when a table in second normal form has a nonkey attribute that depends on another nonkey attribute, a condition known as a **transitive dependency**. Table 4.4 illustrates a third normal form violation.

TABLE 4.3
Second
Normal Form
Violation

<u>Rental-No</u>	<u>Video-No</u>	Due-Date	Title
------------------	-----------------	----------	-------

FIGURE 4.22
Rental and
Video Tables
in Second
Normal Form

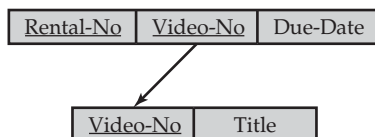


TABLE 4.4
Third Normal
Form Violation

Video-No	Title-No	Vendor	Date-Acquired
----------	----------	--------	---------------

FIGURE 4.23
Tables in
Third Normal
Form

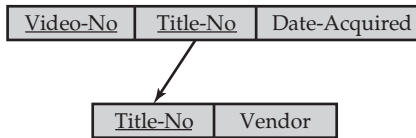


Table 4.4 is in first normal form and automatically is in second normal form because it has only a single attribute primary key. The nonkey attributes, Title-No and Date-Acquired, depend only on the primary key, Video-No. However, because a video title is sold only by the vendor that owns the rights to it, Vendor depends on Title-No, a third normal form violation.

Tables that are not normalized can lead to anomalies, which are problems associated with carrying out operations on the data in the tables. These problems are as follows:

- *Insertion anomaly.* With the video table shown in Table 4.4, GB Video is unable to store information on vendors unless they set up a record for a specific video.
- *Deletion anomaly.* If GB Video disposes of all the videos for a specific title and deletes the records in the video table, the company will lose the information on the vendor.
- *Modification anomaly.* If a vendor sells the rights to a Title-No to another vendor, GB Video must change the vendor data in every row of the Video table that includes the Title-No.

These anomalies can be eliminated by converting the Video table to two smaller tables in third normal form as shown in Figure 4.23.

The fourth, fifth, and sixth steps of normalization eliminate partial functional and transitive dependencies and other anomalies. The database texts listed at the end of this chapter cover these normalization steps.

Structured Query Language

Structured Query Language (SQL) represents a programming language for relational databases that exists at both the logical level and in a number of specific implementations at the physical level. At the logical level, SQL consists of a standard set of commands that appear in all implementations of SQL. Each vendor implementation, for example, MS Access or Oracle Server, also contains a number of vendor-specific features.

In SQL a database consists of a number of related tables. For example, the GB Video database might consist of the Customer, Rental, Rental/Video, Video, and Title tables. SQL provides commands for these languages:

- Data Definition Language (DDL) to create, alter, and drop tables. Some vendors, for example, MS Access, provide a graphical user interface (GUI) that allows

people to create tables without using DDL commands. However, MS Access also contains the SQL DDL commands that one can use to create tables if desired.

- Data Manipulation Language (DML) to insert, update, modify, and retrieve data into or from tables. Again, some vendors also provide a GUI for these tasks.
- Data Control Language to allow the database administrator to grant and revoke access privileges for a database.

SQL is described in such database texts as Hoffer, 2005b and Post, 2005 and in more specialized SQL programming books such as Pratt, 2003. Chapter 10 contains additional discussion on creating and populating relational tables in MS Access and on retrieving data from tables.

Dimensional Models

While the relational model works well for many applications, some specialized applications use a variation called the **dimensional model**. The dimensional modeling principle derives from research by E. F. Codd at about the same time as his work on relational databases. Dimensional models strive to maximize user understanding and ease of retrieval. The basic idea is to use **static data**, which is data generated by operations that will not change over the period of use, to identify problems or discover opportunities in time to take advantage of the information. The users of the data in the dimensional model query the data to gain insight for strategic and tactical decisions.

The typical dimensional model-based data structure is a **data mart**. Data marts are designed around a central fact table that contains numeric values for analysis and dimension tables for keeping track of properties that determine categories and groupings. The process of converting a relational schema into a data mart proceeds as follows:

- *Select a table that corresponds to an associative entity as the central fact table for the mart.* If there is more than one table that corresponds to an associative entity, the system probably will generate more than one potential data mart.
- *The tables that relate both directly and indirectly to the central fact table are dimension tables.*
- *Add the primary keys from all the dimension tables as foreign keys in the central fact table.* In a data mart, adding the foreign key for all the related tables makes for a simpler query structure. This action probably denormalizes the central fact table. That means it creates a third normal form violation; however, it simplifies the structure and operation of the data mart.

The data mart model supplies the framework for creating a **data warehouse** for an organization. Post (2005) identifies some challenges to the effective creation and use of a data warehouse. These are to

- Set up a system to collect and clean the data.
- Obtain acceptable query performance for literally millions or even billions of rows of data.
- Create or procure tools to analyze the data.

The data in the data warehouse often come from capturing data from the ongoing transaction systems that support the operational activities of the organization. For instance, Wal-Mart keeps all sales register checkout information in a data warehouse for over a year and uses the data to learn about customer purchasing habits. The data mart model structure of a single table allows rapid querying of very large sets of data. Sometimes, organizations apply **data mining** techniques to obtain information from the data in the warehouse. Data mining techniques are special programs used to drill down into large bodies of data to extract patterns.

Chapter 11 contains an example of creating a data mart model for a GB Video data warehouse. GB Video management might want to know answers to such questions as the most popular videos by zip code, rental patterns by month, largest customers by sales volume, customers classified by the video type they prefer, and so on. When personnel in purchasing, marketing, or other people have questions, they can consult the data warehouse for insight. The organization also can share selected data with vendors.

Summary

Data and data models affect all aspects of the lives of people. For thousands of years, people have communicated with each other using data: visual and/or verbal symbols. Many information systems serve a primary role of collecting, manipulating, storing, and presenting data. In other words, data play a central role in many information systems. Until several decades ago, information systems often were known as data processing systems.

All information systems use a data model of some kind, that is, some specifications or structures for the data included in the system. The analyst may think about the data and incorporate a data structure (a data model) in the program. Non-IT people often prefer a narrative data model, which provides a text or natural language description of the data in the system. Data modeling concepts and tools discussed in this chapter include entity relationship diagrams (ERDs), conceptual data models (CDMs), enterprise data models (EDMs), and logical data models.

Rules for preparing ERDs include

- *A thing is an entity only if the organization wishes to keep data about the thing.*
- *An entity must contain more than one instance.*
- *Every entity must have an attribute or a set of attributes that serve as a primary key for the entity and the primary key must be unique, that is, a specific value for a key can appear only once in all instances of the entity.*
- *A primary key can be a set of attributes or a composite attribute.*
- *An entity must have two or more attributes.*
- *A relationship must represent a situation that actually exists or the client wants to exist in the organization.*
- *Every entity in an ERD must be linked either directly or indirectly to every other entity in the diagram by one or more relationships.*
- *Every component on an ERD must have a unique name and/or label.*

Rules for supertype/subtype entities include

- *A supertype entity has one or more subtype entities.*
- *The supertype and subtype entities are linked by one-to-one relationships.*
- *Each subtype entity may participate in additional relationships with other entities.*
- *The attributes common to all the subtypes are the attributes of the supertype.*
- *Each subtype may have additional attributes.*

Rules for simplified, reduced-form ERDs include

- *Omit the relationship diamonds but keep a naming phrase for the relationship.*
- *Eliminate the ovals for attributes, list attributes in the entity box, and place the entity name at the top of the box in bold print.*
- *Replace composite attributes with the component attributes.*
- *Replace all the many-to-many relationships by an associative entity using the standard entity symbol.*
- *Replace multivalued attributes with an entity.*

The following rules apply to enterprise data models (EDMs):

- *All of the strong or regular entities that play a role in the entire organization or the organizational area under study are included. Weak, reference, and multivalued attribute entities are omitted.*
- *Relationships are shown with maximum cardinalities, but without minimum cardinalities. Verbs or phrases to describe the relationship are included, but relationship diamonds are omitted.*
- *Many-to-many relationships are acceptable.*
- *Attributes of the entities are not shown.*
- *Associative entities are not included unless they represent an important “thing” in the organization.*

Logical data models translate conceptual data models into a specific data storage structure. Today, the most common logical model for data storage is the relational model. A large number of physical database implementations use the relational model and many, if not most, new applications use relational databases. The relational model retains many of the concepts of entity relationship models but adds additional structure. Data are stored in two-dimensional tables containing rows and columns. Each column represents an attribute and rows correspond to entity instances. A relationship is implemented in the relational model by inserting a foreign key column in one table that references the primary key of the related table.

A relational schema is a graphical representation of the structure of the tables and the relationships between the tables for a relational model. Representations for relational schema that are in common use include column heading, set notation, and box schema. In all three schema, relationships are represented graphically by a referential integrity arrow or line from the foreign key column name

to the primary key column name in the related table. The structure in a relational schema comes from the table concept and from such rules as those listed below.

- *Every table name and the full name of every column must be unique.*
- *A column must have a single value for each row.*
- *The meaning of a column is determined only by the name.*
- *A row is defined only by the content of the information in the row.*
- *The content of each row must be unique.*
- *The data type and format for data in each column must be the same across all rows.*
- *A foreign key column may have any unique full name.*
- *For one-to-many relationships, the foreign key always goes in the table on the many side of the relationship.*
- *Relational tables do not allow many-to-many relationships.*
- *The data type of the foreign key must match the data type of the corresponding primary key.*

The steps in the process of converting a conceptual ERD to a relational schema are

1. Convert all the many-to-many relationships, if any, to associative entities.
2. Convert all multivalued attributes, if any, to entities.
3. Convert every entity (regular, weak, or associative) to a table with column headings that correspond to the attributes of the entity.
4. For *every* relationship, add a foreign key to the table that corresponds to the entity on the many side of the relationship.
5. Add a referential integrity arrow *from* each foreign key *to* the primary key of the entity on the one side of the relationship.

Relational schema for unary relationships appear different than for binary relationships because only one table is involved. However, the rules for converting an ERD representation to a relational schema remain the same for all degrees of relationship.

Normalization addresses the improvement of a logical data design to avoid possible problems with data duplication and with deletion and updating of data. The normalization process converts an un-normalized relation or table into two or more smaller, normalized tables. The normalization process consists of six steps that successively transform a relation into first, second, third, Boyce/Codd, fourth, and fifth normal forms.

SQL represents a programming language for relational databases that exists at both the logical level and in a number of specific implementations at the physical level. At the logical level, SQL consists of a standard set of commands that appear in all implementations of SQL. Each vendor implementation, for example, MS Access or Oracle Server, also contains a number of vendor-specific features.

While the relational model works well for many applications, some specialized applications use a variation called the dimensional model. Dimensional

models use static data—data generated by operations that will not change over the period of use, to identify problems, or to discover opportunities in time to take advantage of the information. Dimensional models known as data marts are designed around a central fact table that contains numeric values for analysis and dimension tables for keeping track of properties that determine categories and groupings. The data mart model supplies the framework for creating a data warehouse for an organization. Sometimes, organizations apply data mining techniques to obtain information from the data in the warehouse.

Key Terms

associative entity, 125	entity relationship model (ER model), 115	primary key, 117
attribute, 117	foreign key, 137	referential integrity, 140
binary relationship, 126	functional dependency, 144	relational model, 136
primary key, candidate class, 117	instance, 117	relational schema, 140
composite attribute, 120	logical data model, 121	relationship, 117
composite primary key, 123	mandatory relationship, 128	simplified, reduced-form ERD (SERD), 130
conceptual data model (CDM), 121	many-to-many relationships, 120	structured query language (SQL), 145
data mart, 146	maximum cardinality, 120	static data, 146
data mining, 147	metadata, 133	subtype, 129
data warehouse, 146	minimum cardinality, 128	supertype, 129
derived attribute, 121	multivalued attribute, 124	table, 137
dimensional model, 146	normalization, 144	ternary relationship, 126
enterprise data model (EDM), 133	one-to-many relationship, 120	transitive dependency, 144
entity, 115	one-to-one relationship, 120	unary relationship, 126
entity relationship diagram (ERD), 114	optional relationship, 128	unique key, 137
	physical data model, 121	weak entity, 124

Review Questions

1. Name some symbols that would convey meaning as data.
2. Define the three technology levels of data models and explain how they are used.
3. When designing an entity relationship model, what questions should the designer address?
4. Why must an entity contain more than one instance?
5. Why should an entity contain more than one attribute?
6. Describe a situation that involves multivalued attributes.
7. Describe a situation where an analyst might need to use a weak entity.
8. Describe the various types of relationships that may appear in an ERD.
9. What is an associative entity? When is the function or role of an associative entity?
10. What is the function of enterprise data models?
11. How do the rules for an EDM differ from those of an ERD?
12. Provide examples of supertype and subtype relationships.
13. Give an example of a unary relationship.

14. Give an example of a ternary relationship.
15. Explain column heading, set notation, and box schema.
16. What is the purpose of a referential integrity arrow or line?
17. Explain how to convert an ERD into a relational schema.
18. You are designing a relational database and accidentally forget to put in an attribute. Is adding attribute to a table a major task? Do the existing programs need to be changed?
19. Why might an analyst normalize the tables in a relational database?
20. What is SQL?
21. How do dimensional models differ from other relational databases?
22. Why would an organization use a data warehouse?

Critical Thinking Exercises

Individual Exercises

For the problems below, complete one or more of parts a, b, c, and d.

- a. Draw an EDM.
 - b. Draw the ERD. Show the primary key for each entity. Show minimum and maximum cardinality for all relationships.
 - c. Convert the ERD to a relational schema using column heading, set notation, or box schema. Include appropriate referential integrity arrows or lines. Include the one and many symbols for box schema.
 - d. Convert the relational schema in part c to third normal form if it is not already in third normal form.
1. The director of a bowling tournament needs a database to connect PLAYERS with MATCHES. The database should contain data on PLAYER-NAME, PLAYER-PHONE, GAME-TIME, LANE-NUMBER, and SCORE for each player.
 2. A car rental company defines customers as business, leisure, and urban. They want to keep name, address, and home phone on all renters, business name and business phone for business customers; leisure customers need destination(s); and urban customers need insurance company and expected numbers of days. When a customer arrives at an office they are assigned a vehicle (VIN, Make, Color, License) and a rate depending on how they reserved the vehicle. When they return the vehicle the agency records days rented and number of miles. Each office owns the vehicles it rents. The database keeps location, manager, and owned vehicles.
 3. Sooner Software Company wishes to sell software products to small businesses. Each product in inventory is identified by a PRODUCT-NUMBER and has attributes PRODUCT-NAME and PRICE. Sooner employs SALESMEN (SALESMAN-ID, SALESMAN-NAME, SALESMAN-PHONE) to promote the products. INTERNAL-SALESMEN have a SALARY and a dollar QUOTA of software to sell. COMMISSIONED-SALESMEN have a RATE. Each SALE to a customer is recorded on a sales slip containing RECEIPT-NUMBER, DATE, SALESMAN-ID, and a list of products with PRODUCT-NUMBER and QUANTITY-SOLD. All products are sold at full PRICE. Each week the company adds up all sales and produces a SALES-REPORT that calculates sales and remaining quota for the internal salesmen and total commission earned for the commissioned salesmen.
 4. A business school wants a database to manage ticket sales for its business conference. The college collects NAME, ADDRESS, and PHONE from all REGISTRANTS. Corporate registrants give their COMPANY and TITLE; Faculty give their DEPARTMENT; and

Students provide their YEAR-IN-SCHOOL and their MAJOR. The Dean also wants to record the COMPANY-NAME, COMPANY-ADDRESS, and DONATION-AMOUNT for all companies having corporate participants. When they arrive, each participant signs up for up to three afternoon SESSIONS for a specific SESSION-TOPIC, SESSION-LEADER, TIME, and ROOM.

5. The city athletic league is forming baseball teams for the spring. Coaches register teams by providing a roster of players and coaches for their team. The roster includes team name and age group, along with first and last names, addresses, and phone numbers for players and coaches. It indicates the head coach's years of experience as well. A player can play on only one team but a coach may coach teams in several different age groups. Once registration is complete the registrar prints a list of teams and coaches for the scheduler to produce schedules. Head coaches record Name, Address, Phone, and Experience. Assistant coaches only record Name and Phone.
6. An IS program wants a database to track potential donations. Donors are either individuals or corporations. For corporate donors record Company Name, Contact Person, Contact Phone. For individuals keep Contact Person, Contact Phone, and Year Graduated. The MIS program identifies a number of Accounts that can be contributed to such as the DPMA Scholarship Account and the MIS Foundation and records Account Name, Account Number, and Account Type (general, scholarship, restricted, etc.). Each donor can donate to one or more accounts each year, and the MIS program wishes to keep track of each year's donation by donor and amount donated to each account.
7. A service club wants to set up a babysitting service. Members sign up for nights that they can babysit. When a customer calls in with a request the project coordinator calls up a Sitter Screen on the club computer that displays which members can sit at given times. He then calls one of the sitters. When the sitter agrees to an engagement, the coordinator enters the customer ID to confirm the engagement between customer and member. Each week he prints a Weekly Summary to show how much each customer owes and how much each member has earned.

Group Exercises

For the problems below, draw the EDM and conceptual ERD, for the problem. Convert the ERD to a relational schema in third normal form.

1. Problem 4 in Chapter 3.
2. A business school is establishing a new Masters degree program in telecommunications and needs to track industry SPONSORS as well as STUDENTS. The director needs NAME, ADDRESS, PHONE, and COMPANY for all contacts. For SPONSORS the director also needs POSITION and COMMITTEES to indicate the area of interest and committees that the individual supports. Students have an ADMISSION STATUS to indicate their enrollment status. The director also wants to maintain a CONTACT LOG for all students to indicate DATE, CONTACT-TYPE, and conversation SUMMARY for contacts to students. Finally the director keeps track of COURSES by COURSE#, DESCRIPTION, and DATE-OFFERED to track enrollment and grades for each of the students.
3. You are building a database for a political campaign. The campaign office starts by building a list of workers with their name and phone number. Workers are either paid or volunteer; paid workers need pay rate and hours worked each week, volunteers need business phone and number of hours available. Each worker is on one or more committees. Committees have Name and Total Budget as attributes; the database also identifies the chairman who can be paid or can volunteer. The campaign maintains

an inventory of supplies for events. When a committee plans an activity, the chairman contacts the volunteers, orders supplies (type, price, and quantity) from the inventory, and supervises the event. Each week the chairman writes a report to the campaign manager detailing the people who participated, the supplies used, and the success of the week's activity.

4. The Premier Products Company is a wholesale hardware company that provides products to customers. Each customer is served by a salesman who processes orders. The salesman is paid from commissions earned on each customer order. A customer places an order by calling the company and contacting the salesman. The salesman records the ordering person, products, and quantity ordered. In the Premier Products Company each salesman is paid from commissions earned on each customer order. A customer places an order by calling the company and contacting the salesman. The salesman records the ordering person, products, and quantity ordered. The order consists of Customer data, Salesman data, and a list of products, price, and quantity for the products that the customer wants delivered.
 5. Construct a data mart design for the rental data generated in Group Exercise 1 above.
- Chen, P. P-S. "The Entity Relationship Model—Toward a Unified View of Data." *ACM Transactions on Database Systems*, March 1976, pp. 9–36.

References

- Codd, E. F. "A Relational Model of Data for Large Relational Databases." *Communications of the ACM*, June 1970.
- Hoffer, Jeffrey A.; Joey F. George; and Joseph S. Valacich. *Modern Systems Analysis and Design*, 4th ed. Upper Saddle River, NJ: Prentice Hall, 2005a.
- Hoffer, Jeffrey A.; M. B. Prescott; and F. R. McFadden. *Modern Database Management*, 7th ed. Upper Saddle River, NJ: Prentice Hall, 2005b.
- Post, Gerald V. *Database Management Systems*, 3rd ed. New York: McGraw-Hill/Irwin, 2005.
- Pratt, Philip J. *A Guide to SQL*, 6th ed. Boston, MA: Course Technology, 2003.
- Whitten, Jeffrey L.; Lonnie D. Bentley; and Kevin C. Dittman. *Systems Analysis and Design Method*. New York: McGraw-Hill/Irwin, 2005.

Chapter Five

Process and Object Modeling

Chapter outline

Introduction

Process Models

Data Flow Diagrams

DFD Symbols

Building a Simple DFD

DFD Rules

Basic Rules

Labeling Rules

Sufficiency Rules

Other Rules and Conventions

Creating Hierarchical DFDs

The Context-Level DFD

The First-Level Explosion DFD

Additional Explosion DFDs

Other Process Models

IPO Charts

Process Hierarchy Charts

Object Models

Use Case Diagrams

Class Diagrams

Sequence Diagrams

Advantages of Object-Oriented Design

Summary

Key Terms

Review Questions

Critical Thinking Exercises

Individual Exercises

Group Exercises

References

INTRODUCTION

Processes provide the engines for an information system. Processes accept data from sources outside the system, place data in and retrieve data from data stores, modify data, transfer data between processes, and send data to users outside the system. In short, processes accomplish all of the work or activity that occurs in a system. This chapter examines process models. The role of information systems models is to add structure to information system analysis and design. Chapter 1 discusses the role of models and the concepts, advantages, and disadvantages of structure. As noted in Chapter 1, the content model views an information system as containing data, process, physical infrastructure, and organizational infrastructure. Chapter 4 discusses data models. The last section of this chapter reviews object models or object-oriented design (OOD). Object models view a system as consisting of objects that communicate with each other and with externals. Objects incorporate both data and process.

From the late 1950s until the late 1970s, people wrote programs and code in whatever way seemed best to them. Some organizations developed structures of their own to guide development, but no standard and generally accepted mechanism existed to provide structure for process knowledge and controls. As programs grew larger and larger, the lack of standard approaches to structuring programs became an increasingly evident problem. Gane and Sarson, 1979, provided the seminal work on a structured approach to systems analysis. Yourdon and Constantine, 1986, followed with work that used different symbols, but followed the basic concept of Gane and Sarson's earlier work. These structured approaches or **process models** allowed analysts to graphically represent the interaction of processes with each other, data stores, data sources, and users following a standard set of rules.

Process models provide abstract representations of real systems that capture some but not all of the structure of the processes in the system in a form that analysts and some clients can understand. Most process models use graphical representations. With process models and the tools that soon followed, analysts could both perform graphical design and track design changes. The process modeling sections of this chapter describe and illustrate several of the better known process models including data flow diagrams. Most textbooks on systems analysis and design discuss process models, for example, see Whitten, 2005 or Hoffer, 2005.

Many of the concepts of object modeling evolved over the past several decades and appear in many applications. Structured object modeling came originally from the work of James Rumbaugh, Ivar Jacobson, and Grady Booch, 1999. These concepts are incorporated in a framework known as the **Unified Modeling Language (UML)**. Version 2.0 of the UML defines 12 diagrams or graphical tools for object-oriented design. The object modeling sections of this chapter describe and illustrate several of the more commonly used object model diagrams including class diagrams. Most systems analysis textbooks provide some information on object-oriented design (OOD). Whitten, 2005, and Hoffer,

2005, contain sections on OOD. Fowler, 2003, provides one of the more understandable guides to UML.

PROCESS MODELS

In a process model, processes play the central role. Processes manage or carry out all of the interactions with each other and with such parts of the system as users and data stores. Process modeling tools that evolved from the structured approach to systems analysis include:

- Data flow diagrams (DFDs)
- Process hierarchy charts (PHCs)
- Input/process/output (IPO) charts








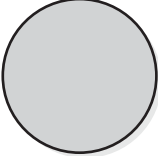

Data flow diagrams, used in Chapters 7 and 8, provide the most detailed representation of processes and their interactions. **Process hierarchy charts**, also called function hierarchy diagrams (FHDs) and used in Chapter 11, show how processes and subprocesses relate to each other. Input/process/output charts represent in table format the relationships between data coming into and going out of the system with a text description of the activities in each process. The following materials discuss each of these process modeling tools along with examples of their use.

DATA FLOW DIAGRAMS

Data flow diagrams provide a detailed, graphical tool for systems analysts to model processes and their interactions in an information system. This tool tracks the flow of information from each external source to a process, where the data are modified and/or sent to a data store, another process, or an external user. Each component in the system—external data source or data user, data flow, process, and data store—has a standard, unique symbol. On the diagram, the data flow symbols show how data flow between the external sources and users, processes, and data stores. Descriptive labels are placed inside or next to each symbol. For example, an external data source symbol might have the label “Customer” to show that the data come from a customer. Detailed textual information about each of the components often appears in a separate metadata table or section of the model. This chapter and Chapters 7 and 8 contain examples of data flow diagrams.

One of the major goals of models is to provide a standardized representation that facilitates communication between systems people working on a project or system and may facilitate communication between analysts and clients. In DFD models, the standardization comes from standard symbols for each component and from standard rules on how to construct the diagrams. The next section illustrates standard symbols used in DFDs. The following section describes standard rules.

FIGURE 5.1
DFD Symbols

Label	Yourdon Symbol	Gane and Sarson Symbol
Data flow		 
Data store		
External		
Process		

DFD Symbols

Figure 5.1 illustrates two possible sets of symbols for building DFDs, the Yourdon symbols and the Gane and Sarson symbols. Note that both use similar symbols with minor variations.

The DFD symbol meanings are

- **Data Flow**—The **data flow** symbol indicates a flow of data from one component to another. For example, data might flow from an external source to a process. For a data flow symbol, Gane and Sarson use a straight line and Yourdon uses a curved line, both with an arrowhead pointing toward the flow destination. The arrowhead shows the direction of the flow of the data. In DFDs, all flows are unidirectional.
- **Data Store**—The **data store** symbol indicates that data temporarily stop moving or are stored. While data flows show data in motion, that is, moving from one component to another, data stores show data at rest or stationary. Filing cabinets and inboxes serve as physical data stores in manual systems. Computer-based systems use electronic, optical, or magnetic devices to store data. The data store symbols differ slightly: The Yourdon symbol is the double line and the Gane and Sarson symbol is an open-ended rectangle with an internal cross bar.

- **External**—The **external** symbol represents a thing outside the system that sends data to the system or receives data from the system. Externals may be an organization, person, group, device, external data store, or other. An external that provides data to the system is known as a **source**, while an external that receives data from the system is known as a **sink**. The external symbols are the same in both of the methodologies. In early articles and texts, externals sometimes are referred to as “entities.” This term was dropped to avoid confusion with the term *entity* in entity relationship diagram data models.
- **Process**—The **process** symbol signifies that an action is taking place, for example, accepting input data, storing or retrieving data, computing a value, using some data, and so forth. Process symbols differ the most with Yourdon using a circle and Gane and Sarson using a rectangle with rounded corners.

In practice, people sometimes mix symbols, for example, they may use the Gane and Sarson symbols for data flows and the Yourdon symbols for processes, externals, and stores because this is the easiest symbol set to draw in a word processing program. Some organizations, analysts, and authors create their own set of symbols, but regardless of the specific symbol set, the concepts and meanings remain the same. Once a person is familiar with the rules and concepts for DFDs, an examination of a DFD quickly reveals the symbol set in use. This chapter and the rest of this textbook use the Gane and Sarson symbols.

Building a Simple DFD

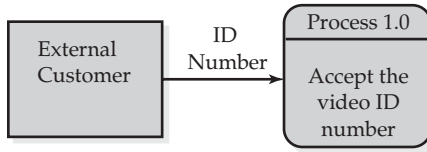
To build a DFD, the analyst combines the symbols to represent the operation of an existing or proposed system. The analyst examines an existing system or thinks about a proposed system to identify the externals, processes, and data stores and the data flows that link them. Some analysts find it useful to write out a description of how the system works; others, particularly analysts with experience, can just look at or think about the system and translate it to a DFD. Such word processing languages as MS Word contain the symbols and features needed to draw DFDs. MS Visio and a number of CASE tools provide features that simplify drawing DFDs. Some of the DFDs in this text originally were drawn with Word; others were drawn with a CASE tool called Visible Analyst.

Analysts follow different paths to create DFDs for a system. Some analysts like to start by identifying all of the major processes since processes form the central focus of process models. Other analysts like to use a building block approach, that is, to identify the components that interact with each process. Consider the following narrative for a simplified version of the GB Video return process discussed in Chapter 7.

When a customer returns a video, a clerk retrieves the rental form for the rental from the file, records the return date on the rental form, and returns the form to the file.

Using a building block approach, the analyst can start by looking at the first block of the video return. The customer (an external) supplies data on the ID number of the video that is returned (a data flow) to a process that accepts the

FIGURE 5.2
First Block of
the Return
DFD

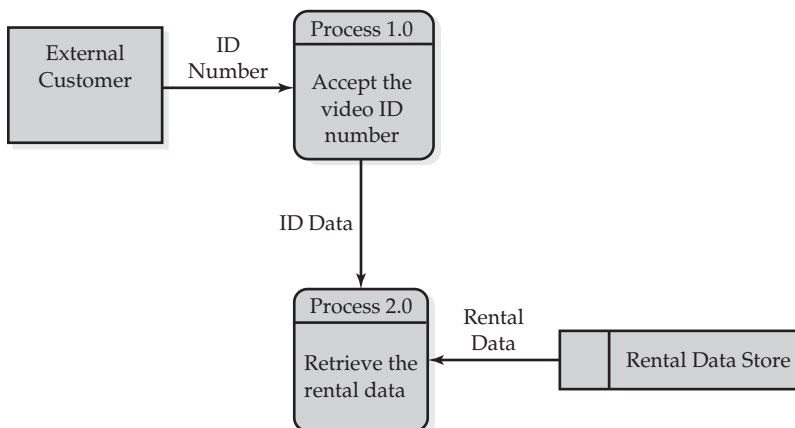


input data. To start the DFD, the analyst can take the symbols for an external, data flow, and process and arrange them as shown in Figure 5.2. The arrow shows that the data flow goes from the customer to the process. Also note that the external, data flow, and process all have names, labels, or descriptions that help the viewer understand the activity that the diagram represents.

DFDs reflect the conceptual view of data and process in the system independent of the physical mechanisms that carry the data or perform the processes. The customer may hand the tape to a clerk who looks at the tape to get the ID number. But conceptually, the DFD shows the customer supplying the video ID number for the return because the customer returns the tape and the tape contains the data, which is the ID number. The data flow reflects the flow of the data, the ID number, not the physical flow of the video. In this case, the ID Number data flow from the customer to the “Accept the video ID number” process.

After the clerk accepts the video ID number from the customer, the clerk uses the ID number to retrieve the appropriate rental form for the tape or DVD (a data retrieval process). The rental forms in the file hold data from the time of the rental until the return; this set of forms is a data store. This information describes a second building block for the DFD consisting of another process, “Retrieve the rental data,” a data store, “Rental Data Store,” and a data flow, “Rental Data.” Because the DFD provides the conceptual rather than the physical model of the system, the process is called “Retrieve the rental data,” not “Retrieve the rental form.” The data flow and data store names reflect the same conceptual view. The expanded diagram with the second building block added appears in Figure 5.3.

FIGURE 5.3
First and
Second Blocks
of the Return
DFD

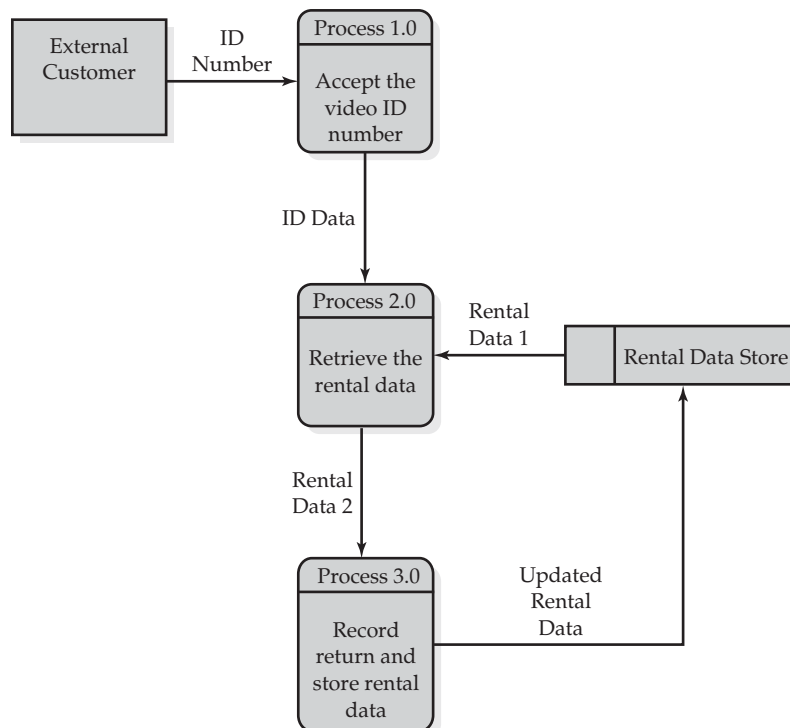


To expand the diagram, the analyst again selects the symbols for the retrieval process, the rental data store, and the rental data flow and arranges them on the diagram. The analyst then notes that the first and second parts of the diagram do not connect. Something must be missing. Indeed, to retrieve the appropriate rental data, process 2.0 must know the ID number of the video. To reflect this requirement, the analyst adds a data flow for the ID data from process 1.0 to process 2.0. This flow must occur for the system to work. Note that the Rental Data flow arrow goes from the store to the process, the correct representation of a retrieval action. In a retrieval from a data store, the process pulls or fetches the data from the store, that is, the data flows from the store to the process. The clerk who is doing all the work in this example does not appear on the diagram because the clerk is part of the system, not an external. The clerk in this case is viewed as a physical mechanism that carries out processes, some of which the computer probably will perform in an automated system.

Two tasks remain for the third and final building block. The clerk must record the day and time that the video was returned and then put the rental data back in the Rental Data Store. The process “Record return and store rental data” is assumed to know today’s date/time; the clerk knows the date/time and computers use an internal clock to generate today’s date/time. Figure 5.4 shows the completed DFD for the return of a video.

The arrow running from process 3.0 to the data store is the correct representation of a data storage action; the updated rental data flows from the process

FIGURE 5.4
The DFD for
Video Return



to the store. Once again, the analyst must add a data flow from process 2.0 to process 3.0 to make the system work. In order to store the updated rental data in the Rental Data Store, process 3.0 must get the original rental data from somewhere, in this case from process 2.0 as shown on Figure 5.3. Note that a one-sentence description of the return generates a DFD with a number of components. Note also that the DFD describes what is happening more specifically and clearly than the text sentence, the traditional value of adding structure.

While the symbols in Figure 5.4 correctly represent the contents of the system, the figure also illustrates a common DFD problem. Where did the original data in the Rental Data Store come from? The original rental data was not created by this system; it must have come from another system, that is, from outside the boundary of the return system. In this event, the Rental Data Store actually should appear as an external—an external data store that can act both as a source and a sink for data. This issue tends to come up frequently today. With modern database technologies, several or many systems often store and retrieve from the same data stores.

DFD Rules

For DFDs to realize their value as an analysis and communication tool, analysts should follow a common set of rules or procedures to construct them. Ideally, or if the rules were complete, different analysts looking at the same system should create identical DFDs. In practice, the rules for DFDs facilitate a standard approach, but each analyst still must make a number of decisions about how to represent the system in DFD format. The rules attempt to achieve the following goals:

- Assure that the DFD model accurately and completely represents the relevant content of the system. DFDs do not claim to model graphically some aspects of a system, for example, the conditional logic of branch points. However, the DFD should accurately model every data flow to and from externals, and to and from the data stores, and between the processes shown in the model.
- Within the limits of the rules, every analyst should model system activities with the same representation. For example, the rules do not specify precisely how to break down or **decompose** a system into modules, but with a given set of modules, every analyst should model the same system with the same representations.

Basic Rules

The basic rules provide the underlying common structure for the DFDs. A number of additional rules deal with specific situations. An examination of the DFD in Figure 5.4 will show that the DFD follows the rules below:

- **Process dominance.** *Every data flow must come from a process or go to a process or both.* Processes perform all the work in the DFD process model. A process is required to accept data from a source, send data to a sink, and store into or retrieve data from a data store. Data may not flow directly between two data stores or between a data store and an external or between two externals.

- **Process completeness.** *Every process must have one or more data flows into the process and one or more data flows out of the process.* When a DFD model shows a data flow into a process and no flow out, clearly something is wrong, that is, the process is incomplete. A process that produces no output adds nothing to the system. When a process has a data flow out and no data flow in, the system probably contains a design error; in other words, the process receives no input data with which to generate an output flow. (Some special exceptions exist, for example, computer subroutines that generate sequential customer numbers, or today's date are processes with output flows but no input flows of a normal kind. On DFDs, these subroutines generally appear only as part of another process and thus their output flows seldom appear on the DFD to trigger a rule violation.)
- **Data store completeness.** *Every data store must have one or more data flows into the store and one or more data flows out of the store.* If a data store has only output flows, then the system contains an error or some other system must be storing the data into the data store, that is, the data store actually is an external. If a data store has only input flows, then either the design is incorrect or some other system uses the data; the data store actually is an external.

Labeling Rules

Labeling rules define how **labels** and descriptions are generated for data flows, data stores, externals, and processes. Labels and descriptions add clarity and resolve ambiguity in DFDs. All the components in Figures 5.2, 5.3, and 5.4 have unique labels that follow the rules follow. At a later stage, the analyst may wish to generate additional information about each of the components on the DFD in a metadata section or table. The unique labels or descriptors for each component allow the analyst to tie the additional information to each specific component in the DFD. Unique labels also allow the programmer to write documentation that ties code modules to specific components on the DFD, that is, Code module 3.1.2 generates the flow of Updated Rental Data from Process 3.1.2 to the Rental Data Store. The rules below present some commonly used conventions for labels. In practice, organizations often establish their own more detailed rules for labels.

- *Every component on a DFD must have a unique label or descriptor.* Some analysts and DFD drawing tools assign sequential, unique ID characters to every component or to some components as part of the label, for example, F1, F2, . . . Fn for flows; D1, D2, . . . Dn for data stores; P1, P2, . . . Pn for processes; and E1, E2, . . . En for externals. Because the IDs are unique, the combination of an ID and a descriptive word phrase always is unique.
- *Data flow labels consist of a word or phrase that describes the data in the flow.* The labels for data flows should be as descriptive as possible, yet short enough to not clutter the chart.
- *Externals have a noun label that describes the source or destination.*
- *Data stores have a noun label.* When the data stores correspond to the entities on an entity relationship diagram (ERD), the data store name is the name of

the corresponding entity. Some analysts use the plural of the entity name; for example, if the entity name is Customer, the data store name is Customers because it contains information about multiple customers.

- *Processes are labeled with a numeric identifier, for example, 1, 2.0, 5.1, 8.1.1, and so on. In addition, processes have a descriptive phrase that varies depending on the type of the process. Process descriptive phrases and numbering schemes are explained in the next section along with the description of process types.*

Sufficiency Rules

The DFD basic and label rules deal with the mechanics of DFDs. The analyst can determine if the rule is satisfied by looking at the diagram. The two rules below are quite different in that they deal with the “correctness” of the representation. A system that follows a correct DFD model will produce the desired results. The analyst cannot tell if the rules are satisfied by looking at the diagram. Instead, the analyst must mentally walk through the processes and flows, with perhaps reference to the metadata descriptions if such descriptions exist, to determine if the rules are satisfied.

- **Process data sufficiency.** *The data flowing into a process must be sufficient to allow the process to (a) perform its tasks and (b) generate the data flowing out of the process.* In the DFDs in Figures 5.3 and 5.4, the analyst had to add data flows from Process 1.0 to 2.0 and from 2.0 to 3.0, in order for processes 2.0 and 3.0 to have enough data to perform their tasks. The analyst could modify Process 3.0 in Figure 5.4 to generate a return receipt data flow to the customer containing the customer’s name if and only if the data flow into the process called Rental Data 2 contains the customer’s name.
- **Data store sufficiency.** *The data placed or stored into a data store must be sufficient to populate any data flows out from the store.* In other words, a process may retrieve from a data store only data stored there by other processes.

Other Rules and Conventions

DFDs provide for a hierarchical structure that starts with a highest level DFD for a system and works downward in ever increasing detail. The hierarchical structure is explained and illustrated in the next section. The additional rules that apply to the DFD hierarchical structure are presented with the discussion of hierarchy levels.

DFDs allow for some alternative representations, for example, split data flows. A **split data flow** is a flow that may begin at one component and go to several other components. For example, several processes may retrieve the same data from a data store. A split flow for this event begins at the data store box as a single line and then splits into several branches that go to the separate process boxes. A split data flow indicates identical information going to different destinations. Use of a split flow in place of several separate flows may relieve clutter

on the DFD. The DFDs later in this chapter use split flows to reduce clutter. Some DFD drawing tools are not designed to handle split flows.

Sometimes DFDs show the flow of control data. Control data tell a process or device what to do. For example, a person or scheduling process may send control data to a reporting process at the end of a month to trigger or start the operation of the reporting process. The external users of the reports never see the control data. If the analyst wishes to show the flow of control data on the DFD, the convention is to create a data flow using a dotted line for the control data. The DFDs in this chapter and in the text do not contain control flows.

Creating Hierarchical DFDs

When an analyst decides to represent a system in a DFD model, the analyst may choose to use a hierarchical approach, especially as systems grow in scope and complexity. The analyst may choose a top-down approach, which means he or she starts with a high-level DFD that represents the entire system as one process and then explodes, breaks down or decomposes the system to create a set of related DFDs that show additional detail. Alternatively, the analyst may use a bottom-up approach. Using this approach, he or she starts with a set of detailed DFDs and then combines the detailed DFDs into higher level DFDs and finally into a single DFD. Or the analyst may use some combination of approaches. In any event, the analyst ends up with a hierarchy of DFDs, some showing overviews of the system and others giving great detail. The combination of overview and detailed DFDs can provide a better understanding of the system than either the detailed or overview DFDs by themselves. This section describes the rules that apply to **hierarchical DFDs** and illustrates the process of creating them using a top-down approach.

The process specifications for the GB Video proposed system appear in Chapter 8. The process specifications are reproduced in Figure 5.5. These specifications provide the content base for the following discussions on creating hierarchical DFDs for a system. The analyst begins the top-down approach by preparing a **context-level DFD**, an overview diagram that represents the entire system as a single process. The analyst then **explodes**, decomposes, or breaks down the single system process into a set of major subprocesses in a first explosion DFD. The analyst may then explode some or all of the processes on the first explosion DFD, and then explode some or all of the processes in the second explosions, and so on until a sufficient level of detail is obtained.

The Context-Level DFD

The purpose of the context-level DFD model is to allow the analyst to describe the interaction between the system and the environment that surrounds the system. The analyst identifies all of the externals and all of the data flows to and from the externals. These data flows define the interaction of the system with the external environment—data flows to and from externals cross the boundary between the system and the environment. As noted, the context-level DFD represents all the processes and subprocesses as a single high-level process. The

FIGURE 5.5 Process Specifications for the GB Video Proposed System

The proposed GB Video Rental/Return system contains the following functions: (1) Member—Create a new member record or update an existing one; (2) Rental—Rent videos; (3) Return—Return videos; and (4) Overdue—Create overdue notices for videos that are overdue. These functions resemble the ones in the current operation. However, using the problem-solving methods described above, the team introduced a number of changes to achieve the customer service and cost goals specified by the client. The client considers all of the functions described in the narrative mandatory.

The customers of GB Video identify the videos that they wish to rent and go to a checkout position. To improve the likelihood that GB rents only to customers who are members, the member option must be selected at the beginning of every rental transaction.

Member

The member process is triggered by a customer request to (1) rent videos and/or (2) become a member. If the customer has and knows the customer number, the number is entered; the system retrieves the record from the Customer data store and displays the customer data. If the customer does not have the number, the customer can provide a telephone number (or a name and zip code). The system tries to retrieve the customer's record from the Customer data store.

If the system is unable to retrieve the customer record or if the customer is not a member, the new member subprocess is initiated. The system will create a new member provided the person has a credit card, telephone, and government-issued ID. The customer supplies the customer data and the data are entered. The system generates a customer number, creates a membership card output, and gives the output to the new member. The system creates a record in the Customer data store for the new member.

Once the appropriate customer record is available, the customer is shown and asked (1) to verify the name, address, telephone, and credit card data; and (2) to report any changes or corrections. This subprocess increases the likelihood that the system will contain current information for the customer. Any change data are entered and the customer data store is updated. When a verified customer record is available and the customer wishes to rent, the member process triggers the rental option and the member data are sent to the rental process. The rental process can be accessed only from the member process; the rental process cannot be accessed directly.

Rental

The rental process is triggered by and only by the member process. The rental process accepts the member data from the member process and generates a new rental transaction number and the rental date. The customer provides the video number and the proposed return date—i.e., the due date. The video number and the due

date are entered into the system. The system retrieves the video data from the Video data store and based on the due date calculates the rental price for each video rental. This process is repeated for each video. After all the videos are processed, the system adds the rental price for each video to get a rental subtotal. The system multiplies the rental subtotal by the tax rate (state sales tax + county sales tax + city sales tax) to get the tax. The total rental cost is the rental subtotal plus the tax.

The payment type—cash, check, or credit card—is supplied by the customer and entered into the system. If payment is by credit card, the credit card data are entered. The system sends the credit card data and the total rental cost to the credit card processor. If the transaction is approved (or is for cash or a check), the system “commits” the rental and line records—i.e., the records are created in the data stores. The rental number, date, clerk number, pay type, and credit card data go to the rental record. The due dates go to the line records for each video.

The system then generates a receipt for the customer. The receipt data includes all of the data supplied by the customer, sent from the member process, and generated by the system during the rental transaction. The system also retrieves the title data from the Title data store and includes the name in the receipt data.

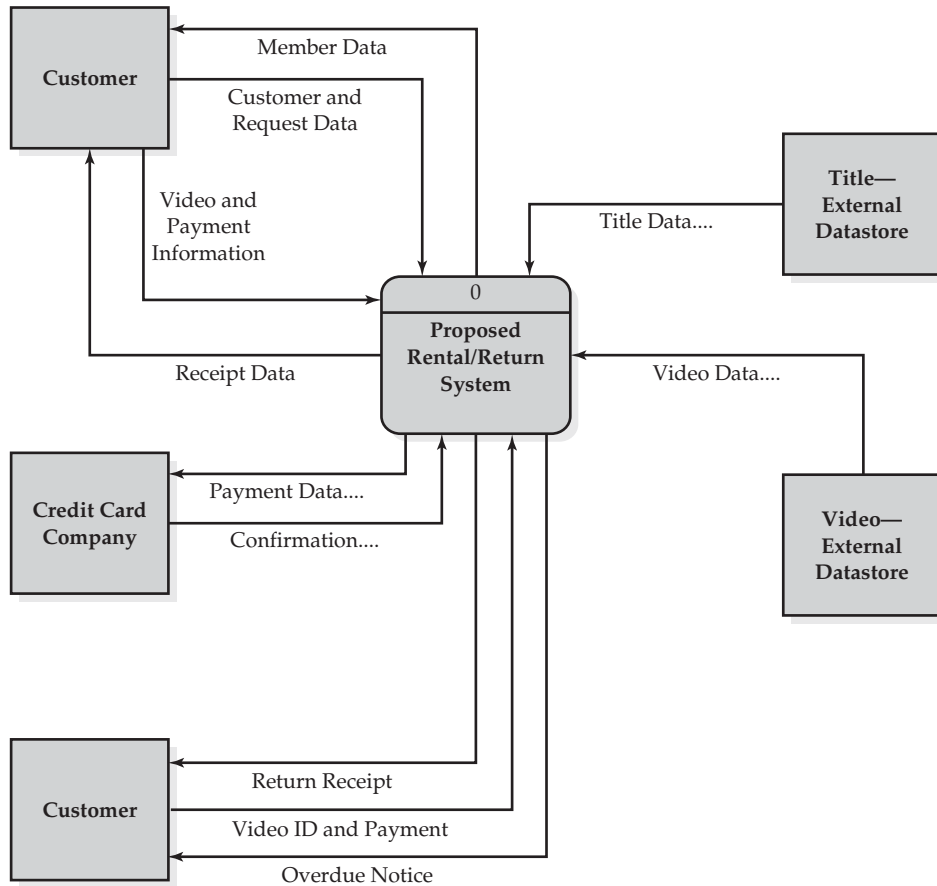
Return

When a video is returned, the video number is entered into the system. The system retrieves the corresponding line and rental records, enters the return date, and calculates overdue charges if any. The customer may charge the overdue fee to the credit card used for the rental or may pay by cash or check. About 95 percent of the time, customers return videos with no payment type input—for example, drop them in a return box. In the absence of customer input on payment, the system retrieves a credit card number from the customer record and processes the overdue charge against the credit card. The system records the charge and payment type in the Line data store. The system may create a return receipt for the customer.

Overdue

After the overnight returns are processed, a clerk instructs the system to run the overdue program, i.e., triggers the overdue function. The system processes the Line records. For each video that is two or more days overdue (i.e., today's date – due date ≥ 2), the system retrieves the rental record for the video, retrieves the customer record for the rental, generates an overdue notice, and sends the notice to the customer. When a video is 14 days overdue, the system retrieves the customer's credit card number from the Customer data store and the video cost from the Video and Title data stores, charges the customer's credit card for the amount of the cost of the video, and sends a notice informing the customer of the charge. If a customer has a complaint about overdue charges, the complaint is handled and processed by Accounting.

FIGURE 5.6
Context-Level
DFD for the
GB Video
Proposed
System



process box is the boundary of the system. Everything inside the box is part of the system; everything outside the box is external. The context-level DFD derived from the GB Video proposed system appears in Figure 5.6.

To prepare the context-level DFD, the analyst looks at the written narrative description or thinks about the proposed system if no written description exists. The process part is easy; the analyst just places a single process box in the center of the diagram with a description in the box that shows the name of the system. On Figure 5.6, the process box contains a numerical label of zero in the space at the top. The label of zero or 0.0 is used to indicate the highest level or root process in a system.

Reading the narrative quickly shows that the customer serves as both an external data source and sink: Customers provide input data flows and receive output data flows. Further reading reveals that payment data flows go to an external credit card processor, and confirmation flows come back into the system from the credit card processor, that is, the credit card company is another external source and sink. The analyst now can add the externals Customer and Credit Card Company to the DFD in Figure 5.6.

Identifying the data flows to and from the externals requires careful reading or thinking. The analyst identifies each flow and assigns a name to it. The data flows associated with customer in the Member and Rental paragraphs of the specifications and the corresponding text that describe the flows are as follows:

- **Customer and Request Data.** “If the customer has and knows the customer number, the number is entered; the system retrieves the record from the customer data store and displays the customer data. If the customer does not have the number, the customer can provide a telephone number (or a name and zip code).”
- **Member Data.** “Once the appropriate customer record is available, the customer is shown and asked (1) to verify the name, address, telephone and credit card data . . .”
- **Receipt Data.** “The system then generates a receipt for the customer. The receipt data include all of the data supplied by the customer, sent from the member process, and generated by the system during the rental transaction. The system also retrieves the title data from the title data store and includes the name in the receipt data.”
- **Video and Payment Information.** “The customer provides the video number and the proposed return date—the due date. The payment type, which can be cash, check, or credit card, is supplied by the customer and entered into the system. If payment is by credit card, the credit card data are entered.”

In similar fashion, the analyst identifies and names the three flows to and from the customer in the Return paragraph (Return Receipt, Overdue Notice, and Video ID and Payment) and the two flows associated with the credit card company (Payment Data and Confirmation). Note that the symbol for the external, Customer, appears twice in Figure 5.6. To allow for shorter, simpler data flow arrows, analysts may place multiple copies of external symbols on the diagram.

The two external data stores shown in Figure 5.6 may cause the analyst some problems. An experienced analyst working at GB Video may know that these two data stores are created by other systems. When this analyst sees data coming from them in the narrative, the analyst may recognize that the data actually come from externals, in this case, from external data stores. An analyst from outside GB Video at first may identify Title and Video as data stores inside the system and only recognize them as externals when they violate the Data Store Completeness rule in the First Explosion DFD. The two stores will have output flows but no input flows. Often, an analyst who starts with a context-level DFD will need to correct it after preparing the first explosion DFD and to correct the first explosion after preparing the second explosions, and so on. System analysis and design are iterative processes.

In summary, the following rules apply to context-level DFDs:

- *All of the processes in the system are represented by a single process box with the system name as the description in the box. The outline of the box represents the boundary of the system.*

- Each and every external in the system must appear on the diagram.
- Each and every data flow in the system to or from an external must appear on the diagram.
- Data stores, as such, must not appear on the diagram; by definition, data stores are internal to the system. Data stores created or used by other systems may appear on the diagram as externals.

The First-Level Explosion DFD

The context-level DFD provides a good overview of the interaction between the system and the external environment, but it does not provide much detail about what goes on inside the system. The act of providing more detail is called “exploding the process.” Exploding a process consists of the following steps:

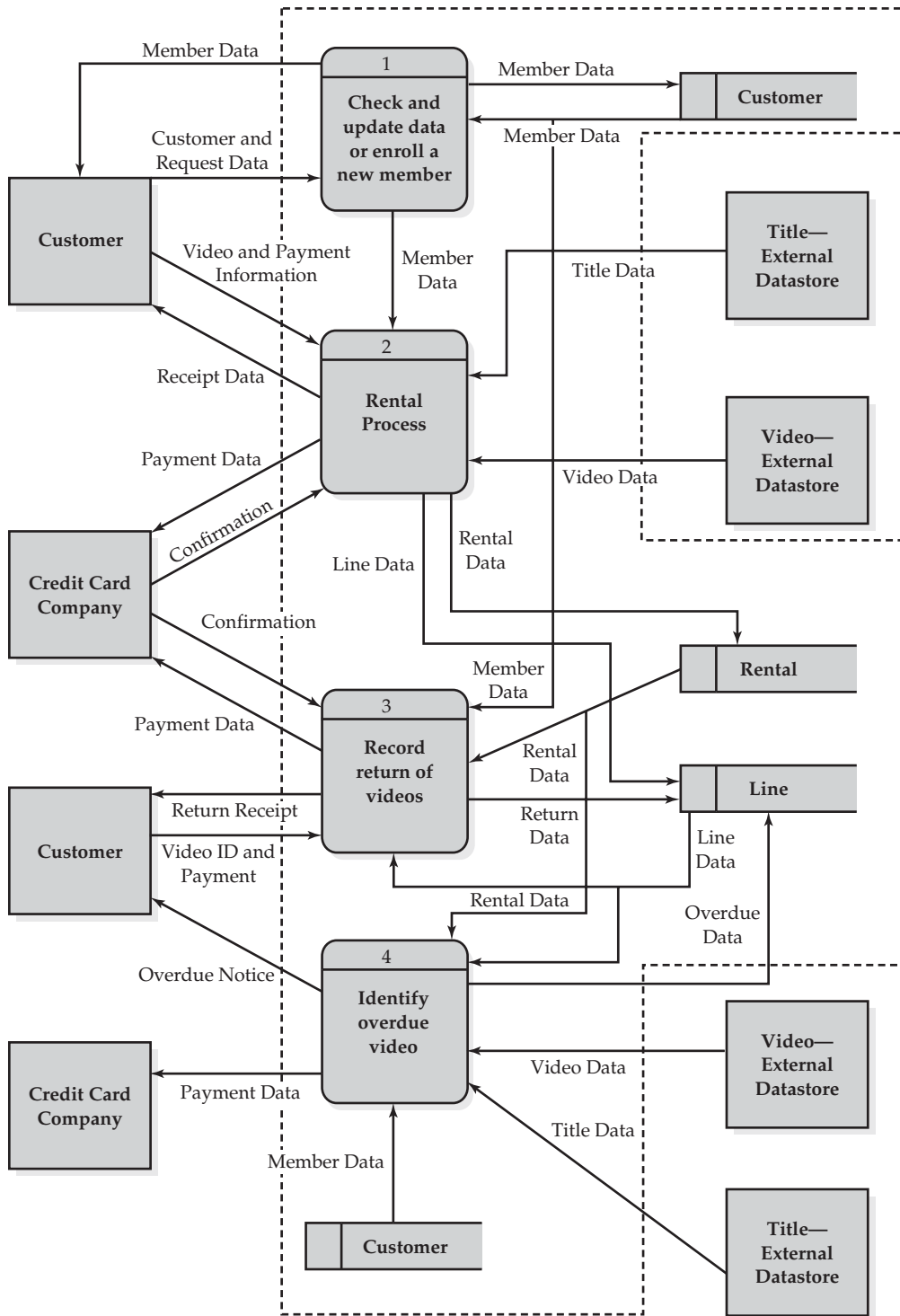
- The analyst divides or decomposes the original process into subprocesses. The new subprocesses are placed on the Explosion DFD.
- All of the data flows into or from the original process are connected to one of the subprocesses on the DFD.
- Data stores used by the subprocesses are placed on the DFD.
- Data flows are added to reflect retrieval from the data stores. Data storage flows are added and must satisfy the data store sufficiency rule.
- Subprocess to subprocess data flows are added as needed to achieve process data sufficiency.

Any process for which the analyst can define subprocesses can be exploded. The explosion of a process can result in two types of processes:

1. An **elementary process (EP)**, primitive process, or **basic function module (BFM)**. In concept, an elementary process consists of only one process with no subprocesses. The analyst cannot explode an EP because it contains no subprocesses. In practice, an EP is the lowest level of detail that the analyst wants on the DFD: The analyst does not want to generate an explosion for an EP. The description for an EP starts with a present-tense active verb and consists of a phrase that describes the action performed by the EP, such as “Identify overdue video.”
2. A **compound process**. In concept, a compound process contains two or more identifiable subprocesses, which may be compound processes or elementary processes. In practice, when a compound process appears on a DFD, an explosion DFD for the process exists. The description of a compound process contains the word *process* at the end of the name, such as Rental Process.

Figure 5.7 shows the first-level explosion of the system process on the context-level DFD in Figure 5.6. Some analysts prefer to start with the first-level explosion DFD and dispense with the context level. Some DFD drawing tools do not include context-level diagrams. Conceptually, the context-level DFD is the root of the hierarchy from which the first-level explosion and all other DFD explosions evolve. Since the context diagram contains only one process, the system process, only one first-level explosion DFD will exist. The dotted line represents

FIGURE 5.7 First Explosion DFD for GB Video Proposed System



the system boundary; the space enclosed by the dotted line corresponds to the process box on the context diagram—all the components inside of the boundary box are part of the system.

Everything that crosses or is outside the boundary box corresponds directly and exactly to the data flow and external components on the context DFD. Every data flow from or to an external on the context diagram must appear on the first explosion DFD. If a new external and/or data flow to or from an external is needed, the new flow and or/new external also must be added to the context-level DFD. To reduce clutter, four flows: Title Data, Video Data, Payment Data, and Confirmation, appear only once on the context-level diagram with names followed by “...” to indicate that they represent multiple flows of the same data. Each of these flows appears several times on the first-level explosion.

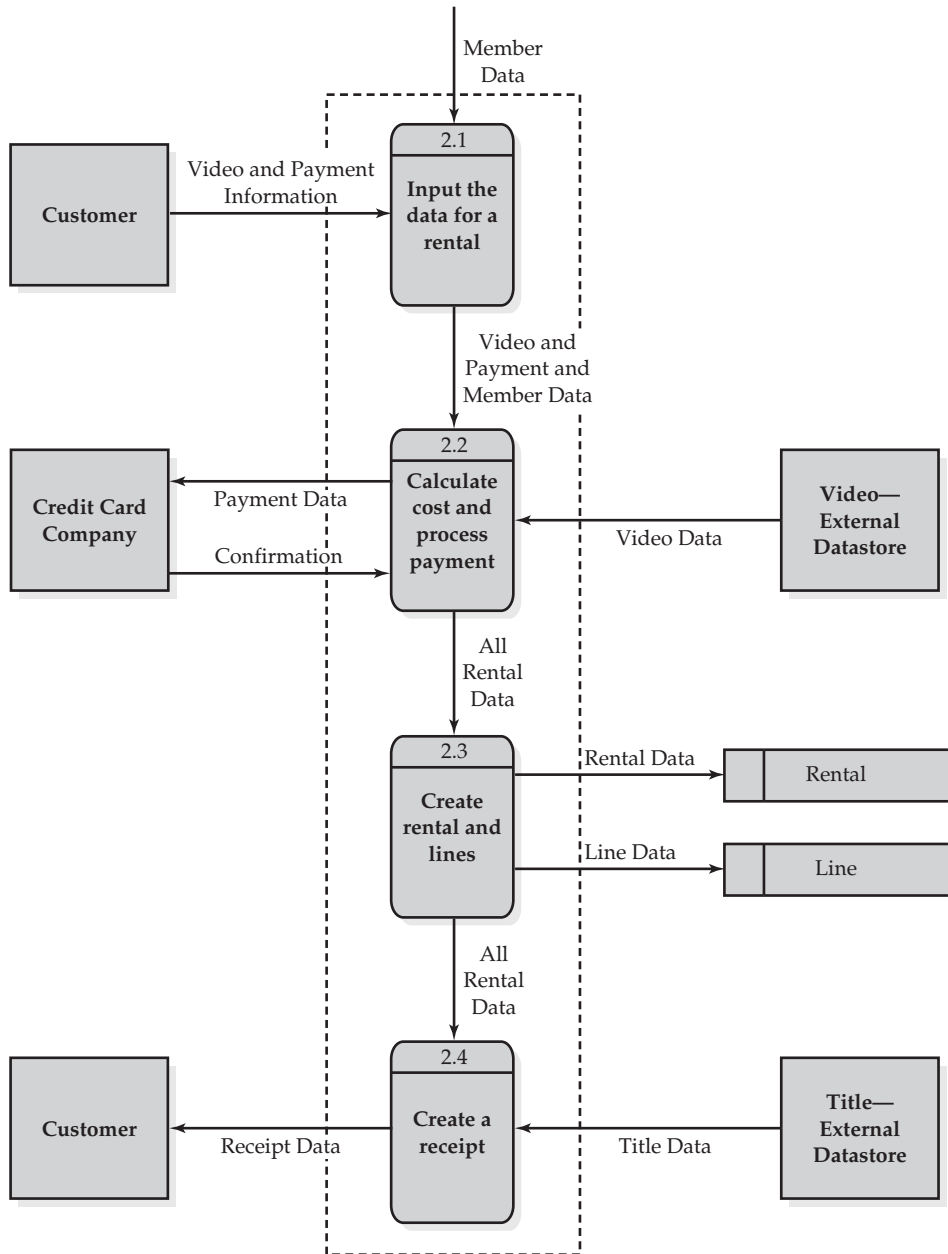
To start adding components inside the boundary box in the DFD of Figure 5.7, the analyst identifies the major subprocesses. Based on the process specifications and/or thinking about the rental/return system, Member, Rental, and Return seem like the appropriate major subprocesses. The specifications also talk about an Overdue process, that is, what to do when tapes or DVDs are overdue for return. These four processes appear in Figure 5.7, three with BFM type descriptions and one, Rental Process, with a compound process description. The description Rental Process tells the viewer that an explosion exists for this process. Each process also receives a number label such as 1, 2, ...n. The analyst connects the external data flows to the appropriate subprocesses. From the specifications and from the ERD when one exists, the analyst can determine that the data stores are Customer, Rental, and Line. Two symbols for the Customer data store appear on the diagram, again for clarity and to keep the chart uncluttered.

The analyst can now add new internal data flows as needed between processes or between processes and data stores. As described in the specifications, Member Data flows from process 1 to process 2. The remaining new flows represent retrievals from data stores (flows from the store to a process) and storage of data (flows from a process to a data store) as described in the process specifications. In DFDs, as in the real world, data are perishable. Because the rental and return processes take place at different times, the rental process cannot send Rental Data and Line Data to the return process. Instead, the Rental Process stores the data in the Line and Rental data stores and the return process (process 3) retrieves the data from the Rental and Line data stores when they are needed.

Additional Explosion DFDs

As noted previously, any process with subprocesses can be exploded. While only one first-level explosion DFD can exist, the analyst can create a second-level DFD for each process on the first level, and then third-level DFDs for processes on the second-level DFDs, and so on. Thus the analyst could create four second-level DFDs for the four processes on Figure 5.7. Because textbook authors and readers have limited time and patience, this chapter contains only one second-level explosion DFD and no third-level explosions.

FIGURE 5.8
Explosion of
Process 2.0



To explode process 2, the analyst again follows the procedure given earlier for explosions. The new second-level explosion of process 2 appears in Figure 5.8. The components inside the dotted boundary box are what exists inside process 2. The newly created subprocesses contain numerical labels of 2.1, 2.2, 2.3, and 2.4. The final integer in the label is assigned sequentially to the new processes

on the DFD. The leading integer in the label separated by a period shows that the new processes are part of process 2. The external flows on this chart match exactly the flows into and from process 2.0 in Figure 5.7. The new process to process flows are required to satisfy the process data sufficiency rule, which means the processes need the data in these flows to perform their actions.

In summary, explosion DFDs must follow all DFD rules as well as the additional rules listed below:

- **Decomposition.** *An explosion must contain at least two subprocesses.*
- **Consistency.** *Every data flow and only data flows into and out of the original process can cross the boundary box of the explosion and connect to one of the subprocesses. Each of these flows must have exactly the same name on both diagrams. Existing flows into the original process cannot disappear, and new flows that cross the boundary box of the explosion cannot appear.*
- **Labeling.** *In an explosion, compound processes receive a noun and adjective description that ends with the word Process. BFM's receive a text phrase that begins with an active, present-tense verb and describes the action performed by the BFM. All processes receive a numeric label. The label for processes in an explosion is the numeric label of the original process followed by a "." and an integer.*

OTHER PROCESS MODELS

Many other process models in addition to DFDs exist. Most computer-assisted software engineering (CASE) tools contain one or more process models, often models tailored to the specific viewpoint of the vendor and designed to work with the vendor's other development tools. The actual code for a system is a highly specific and detailed process model. Program code represents an operational model; the analyst can cause the computer to execute the code and observe whether or not the code model performs as desired. Most process models are nonoperational; they represent precursors to the operational model, the program code.

The remainder of this section describes two other process models: IPO charts and process hierarchy charts. Both of these models bear a relationship to DFDs. Chapter 11 discusses and applies some additional process models.

IPO Charts

The **IPO (input/process/output) chart** is a process model used to study a system by identifying the input and output data flows. IPO charts consist of a table with these three columns:

- The left column contains the external data that flows into the system.
- The center column contains descriptions for every one of the BFM processes in the system.
- The right column contains the external data flows from the system.

A sample IPO chart for the GB Video proposed system appears in Table 5.1. The input and outputs correspond to the external data flows in the context-level

TABLE 5.1
IPO Chart for
GB Video

Input	Process	Output
Customer and Request Data	Check and update data or enroll a new member	Member Data
Video and Payment Information	Input the data for a rental	
Confirmation; Video Data	Calculate cost and process payment	Payment Data
Title Data	Create a receipt	Receipt Data
Video ID and Payment; Confirmation	Record return of videos	Payment Data; Return Receipt
Title Data; Video Data	Identify overdue video	Payment Data; Overdue Notice

DFD. The processes correspond to the BFM processes that appear on the first- and second-level explosion DFDs. Thus, the IPO chart captures some but not all of the information in the hierarchy of DFDs in a compact table format.

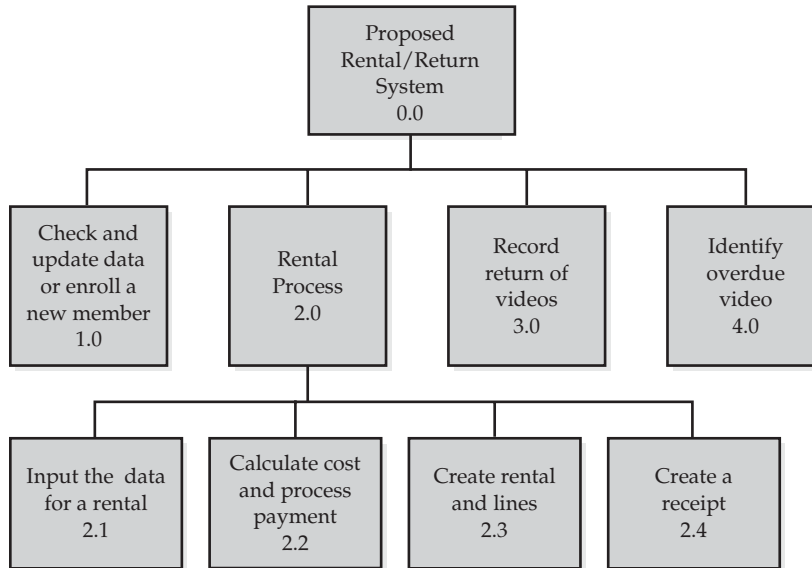
To use the IPO chart as a first step in the analysis or design of a system, the analyst might determine the outputs from the system that the client wants or specifies and enter them as shown in the right-hand column of the table. The analyst then must ascertain which inflows the system will use to produce the required outflows. The last step is to determine the processes associated with each of the inflows and outflows—in other words, what functions accept the input data and what functions produce the outflows for the system. The rows of the chart may appear in any order, not necessarily chronologically. As in all parts of system analysis and design, the construction of IPO charts is an iterative process. The analyst may go through the IPO process several times.

Process Hierarchy Charts

The process hierarchy chart (PHC) or function hierarchy diagram (FHD) is a process modeling tool that graphically displays the hierarchical relationships between processes in a compact format. The design process for a PHC follows the design for hierarchical DFDs. With a top-down approach, the analyst starts with the one process that represents the entire system, the root process. The analyst then decomposes the root process into the major subprocesses and decomposes the new subprocesses as appropriate. Alternatively, the analyst may start by identifying the lowest level processes and build upward or some combination.

A process hierarchy chart for the GB Video Rental/Return System described in Chapter 11 appears in Figure 5.9. The chart corresponds directly to the DFD hierarchy for GB Video developed in the previous section. In the DFDs, the processes appear in three separate diagrams. In the PHC, the processes appear on a single chart that shows their relationships to each other. However, only the processes appear in the PHC; all of the information about externals, data flows, and data stores has disappeared. Process hierarchy charts, for this reason, are referred to as “pure” process models in contrast to DFDs which combine information about processes with information about data.

FIGURE 5.9
Process
Hierarchy
Chart for GB
Video



PHCs use the same process naming and labeling conventions as DFDs. Processes 1.0, 3.0, 4.0, 2.1, 2.2, 2.3, and 2.4 are at the elementary process or BFM level. Process 2.0, as shown by the name and the diagram, has an explosion.

Analysts may use process hierarchy or similar charts as a guide for programmers to write code. When PHCs are used as a coding guide, the analyst tries to decompose each process until the resulting BFM each represent a single block of code consistent with structured programming rules for code blocks. When PHCs are constructed from the bottom up, a primary step in the construction of the chart is deciding how to group the processes. Since all of the elementary processes labeled 2.1, 2.2, 2.3, and 2.4 refer to rental activities, the analyst can group the related processes together as the rental process.

As noted earlier, no data flows, externals, or data stores appear in the PHC. The analyst provides this information in metadata for each data store and process. The metadata for each process describe all of the creation, update, and retrieval actions for data stores, and the data flows to and from externals and between processes. Chapter 11 contains an example of metadata for the processes on the PHC for the GB Video proposed system shown in Figure 5.9.

OBJECT MODELS

Many system analysts and designers tend to think in process model terms; they use a “procedural” approach building systems around a set of procedures or processes. In a typical batch-style procedural structure, a program begins, proceeds sequentially through a series of activities, and ends. As information technology evolved, especially with interactive systems, some designers began to recognize different underlying program structures. These “different” programs had no standard order or sequence for activities; users of the systems determined

the flow of the program. Consider a typical personal computer operating system and the associated word processing and other programs. The program designer has no idea in what order the user will wish to execute the various features available in the operating system and programs.

Designers of interactive programs started thinking and building these systems in what they called objects: self-contained modules that could communicate and interact with many other modules in the overall program. Such languages as Visual Basic appeared with a structure that allowed the programmer to write the code associated with forms and events; for example, accept some input data from a user in a form and put the data in a table, or generate a report. The modules the programmer writes in VB for each event are largely independent, but modules or objects can communicate with each other. The GB Video system builds around a set of events: (1) a request by a person to become a member; (2) a request by a member to rent one or more videos; and (3) the return of a video.

As the evolution of systems design continued, some people realized that a new and different design structure might offer advantages over the DFD/PHC type structures of the procedural model. The new model, Object Oriented Design (OOD), came to have the following significant features:

- With OOD, programs and systems are structured not around data, processes, or events, but around “objects.” **Objects** are things that exist and play a significant role in the actual physical environment of the system. Clients and users think of a system in terms of objects. In the GB Video example, the conceptual-level objects are those that by now seem familiar to the reader: customer, rental, rental-line, and video. At the physical and logical levels of design, other objects appear. For example, the workstation or point of sale screens used by the GB Video clerks become objects in the system OOD at the physical level.
- An object represents an individual thing, for example, a customer object represents a single customer. An **object class** represents all of the objects in a system that are members of the class. The object class Customer represents all the customer objects (instances) in the system.
- Objects contain within them the data and **operations** (processes) required to carry out their desired behavior. All object classes contain standard “CURD” operations to: (1) *create* a new object in the class; (2) *update* and/or (3) *retrieve* data for existing objects; and (4) *delete* objects. The Customer object class or the objects within the class store the data about customers and know how to create an object for a new customer and to update and/or retrieve data for existing customer objects. In other words, the objects in the Customer object class manage the Customer Data Store for the system.

In addition, to the data management behavior standard for all objects, some objects may contain operations for other behavior. The objects in the Rental object class, in addition to managing rental data, must contain an operation to calculate the total charge for a rental, which may consist of adding up the rental charges for one or a number of videos and calculating and adding the

sales tax. In summary, an object is an entity instance on steroids; it contains all the data for the instance and also all the behaviors that apply to the object class.

- Objects interact with each other and with externals by sending and receiving **messages**, such as “Here is some data and/or a task for you,” or “Send me the data about x.” Procedures or operations and data in an object are **encapsulated** or hidden. A rental line object doesn’t know or care how a rental object calculates the total charge for a rental. All a rental line object has to do is send the rental charge for itself when it receives a message to do so. The rental object doesn’t care how the rental line objects store, retrieve, and process data as long as the rental object receives the rental charges back after sending messages to obtain them.

In the early days, every person working on OO design had his or her favorite way to model the system. Three object pioneers, Grady Booch, Ivar Jacobson, and Jim Rumbaugh, pooled their ideas to create the beginnings of a Unified Modeling Language in 1997. The Object Management Group (OMG), a consortium of companies, continued work on the standard and has released a number of additional versions of UML, the most recent of which is version 2.0 (see www.omg.org for specifications and the most current information). Most of the UML definitions are highly technical and written in complex language. While UML provides a standard structure for object-oriented design and development, UML, in common with DFDs, ERDs, and most procedural models, is not a design methodology and does not specify a design process.

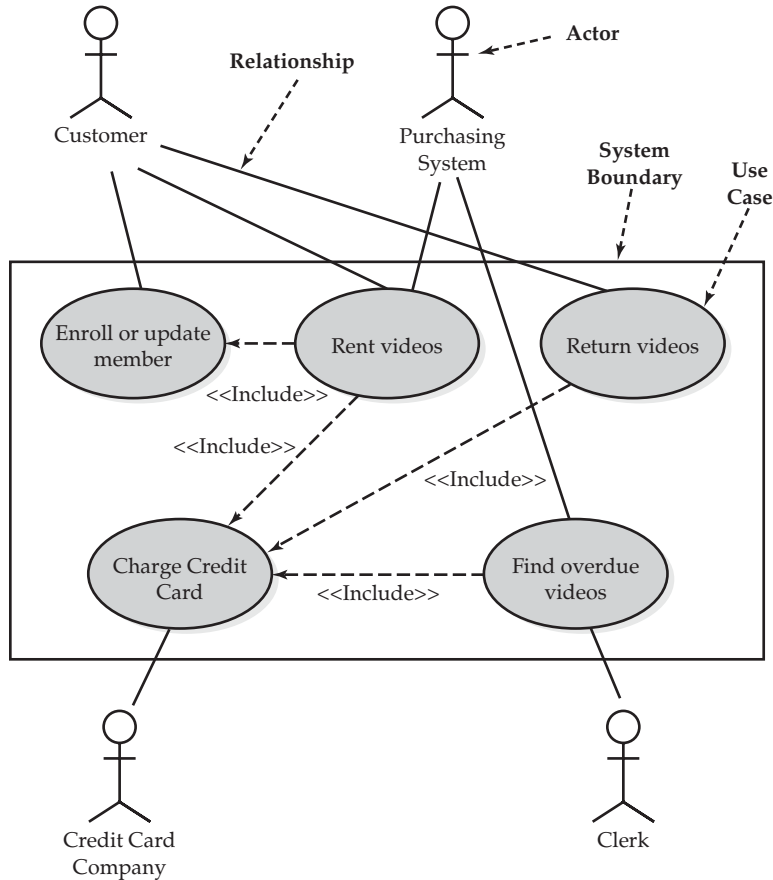
UML version 2.0 specifies 12 diagrams for object-oriented modeling of a system. Four diagrams—class, object, component, and deployment—display the structure or a static view of the system. Five diagrams—use case, sequence, activity, collaboration, and state—display dynamic behavior of the system over time. The other three diagrams are used for model management. Fowler, 2003, describes and explains each of the diagrams. Remember that models are representations of the actual system. Because all systems contain process, data, and infrastructure as discussed in the content model of Chapter 1, some of the UML diagrams, not surprisingly, resemble familiar diagrams from previously discussed data and process models. An object model, data model, and process model of a system all represent parts of the same underlying system, albeit with different focus, symbols, and formats.

The next sections describe and illustrate three of the more commonly used UML diagrams: (1) use case diagrams, (2) class diagrams, and (3) sequence diagrams. Example diagrams from the GB Video proposed system in Chapter 8 illustrate the appearance, content, and use of each diagram.

Use Case Diagrams

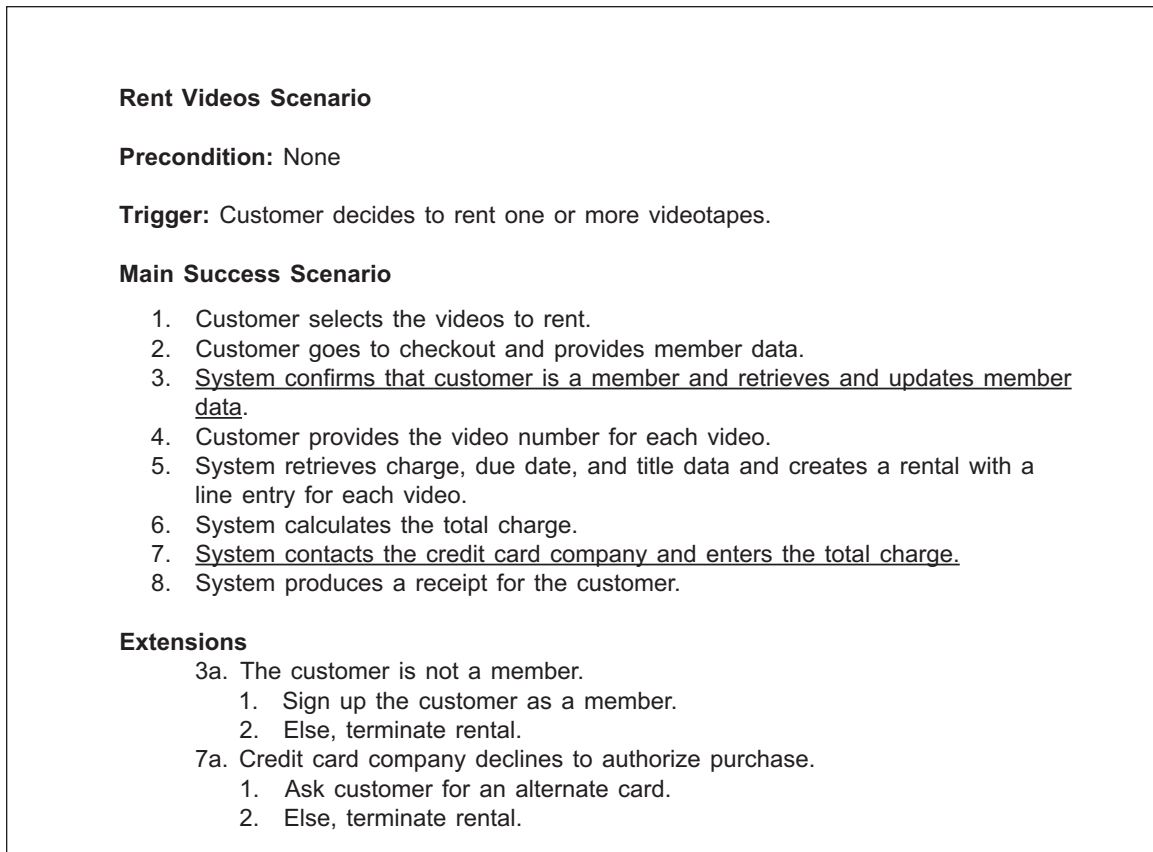
Use case diagrams give the analyst a way to display the interaction of system processes with each other and the external environment. The UML provides a diagram format for use case diagrams but has little to say about the appropriate

FIGURE 5.10
Use Case
Diagram



or correct content. Figure 5.10 shows the use case diagram for the GB Video proposed system. The use case diagram looks like a process model of the system and resembles the first explosion DFD in Figure 5.7. The labels in bold type with dotted line arrows show the components: **actors** (externals in a DFD), **use cases** (processes in a DFD), the *system boundary* (same in a DFD), and the *relationship* lines that connect the actors to the use cases that involve them (vaguely similar to external data and/or control flows in a DFD).

Three of the use cases, Rent Videos, Return Videos, and Find Overdue Videos, all involve a common activity: contact the credit card company and place a charge against the customer's credit card. The use case diagram shows this situation by creating a use case called "Charge Credit Card" and letting the other three use cases make use of or include the Charge Credit Card use case. Rent Videos also includes the Check Member use case. The <<include>> relationship is shown by a labeled arrow with a dashed line. The Rent and Find use cases interact with the purchasing system in that they use data from the Video and Title data stores maintained by Purchasing. Some analysts incorporate additional

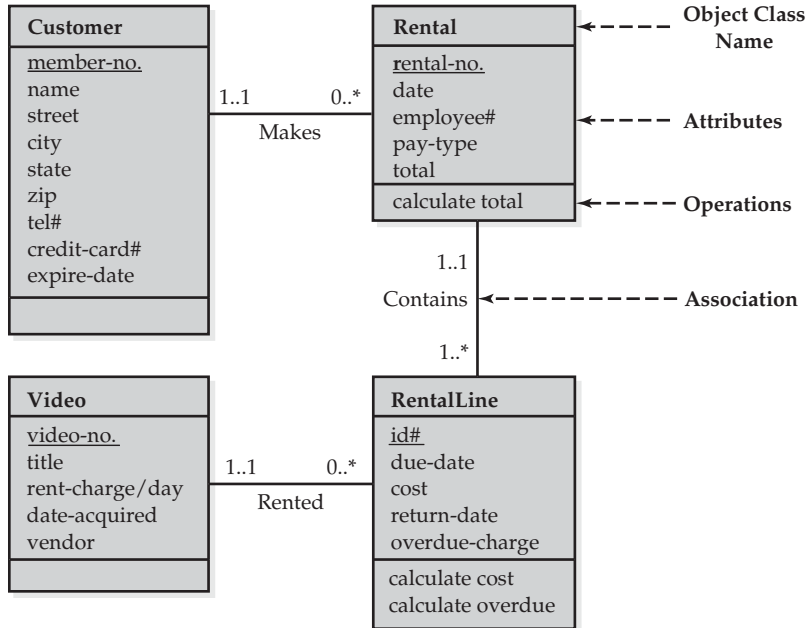
FIGURE 5.11 Scenario for the Rent Videos Use Case

UML-defined features in use case diagrams, but the additional features may add more confusion than understanding. All of the behaviors shown on the use case diagram must appear as behaviors associated with one of the objects in the OOD version of the system.

The most important part of the use case diagram is the metadata for each of the use cases on the diagram. The description of the content of a use case is known in object-speak as a *scenario*. Scenarios resemble pseudocode. Figure 5.11 shows the scenario for the Rent Videos use case in Figure 5.10.

The scenario can have some header information, in this case, a precondition and a trigger. The precondition tells what the system must ensure is true before this use case begins. The trigger specifies the event that starts the operation of the use case. The main success scenario describes the things that happen when all goes well and the end result is a successful rental. The extensions show by line number in the main success scenario, things that can go wrong and prevent a successful rental. The underlining in steps 3 and 7 indicates that these steps include another use case as shown in Figure 5.10.

FIGURE 5.12
Class
Diagram for
GB Video
Proposed
System



Class Diagrams

A **class diagram** shows the static structure of the system and represents the key or central diagram in an object design. The class diagram describes each object class in the system, identifies the data and operations for each object class, and shows the associations between object classes. Class diagrams resemble ERDs but follow a different concept and contain additional information. A simplified conceptual-level class diagram appears in Figure 5.12.

Each rectangle represents an object class. The object class rectangles are divided into three areas by horizontal lines. The top area contains the object class name, the middle area has the **attributes** of the class, and the bottom area shows the operations for the class. Object class names start with uppercase letters; attribute and operation names start with lowercase letters. Video appears as an object class on this diagram because video objects supply data to the rental system. However, the Video object class is controlled by the purchasing system. Other objects in the rental system can ask the video objects to retrieve data but may not ask the video objects to perform any other operations.

A solid line between two object class symbols represents an **association** (similar to a relationship on an ERD). An association may show multiplicity in a format ([min]..[max]) similar to minimum and maximum cardinality on an ERD. The 1..1 symbol next to Customer indicates that each rental object is associated with one (minimum) and only one (maximum) customer object. The 0..* symbol next to Rental indicates that a customer object may be associated with zero (minimum) or many (maximum) instances of rental objects.

Attributes of object classes specify the **properties** of the objects in the class. Attributes of objects resemble attributes for entities, but the diagram may display such other properties in addition to the attribute name as:

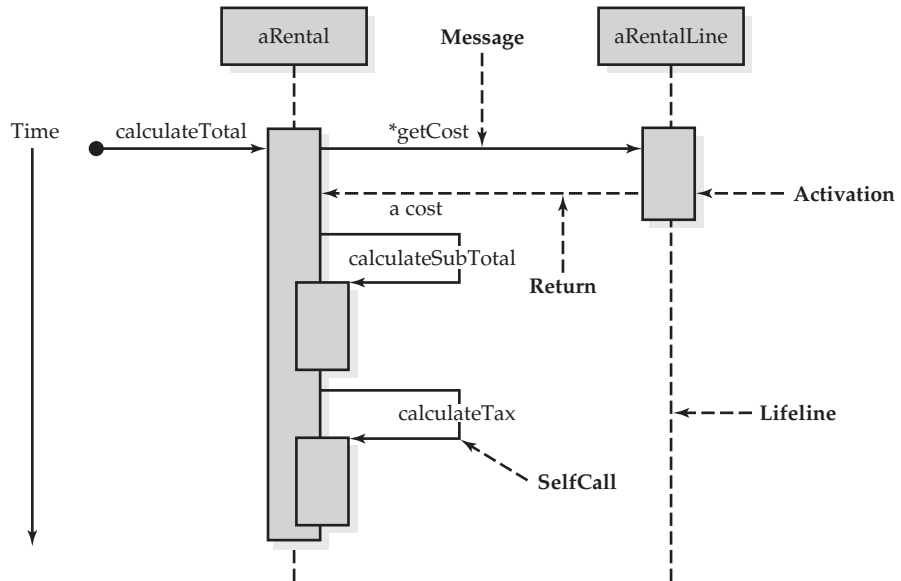
- *Visibility*. Tells if the attribute is public, visible to other objects (similar to a global variable), or is private, for internal use only by the owning object (similar to a local variable).
- *Type*. The data type: Date, String, Integer, Money, etc.
- *Multiplicity*. Similar to minimum and maximum cardinality.
- *Default*. The value of the attribute when a new instance of the object is created when the creation process does not specify a value (similar to a null value).
- *Property string*. Additional properties, for example, {readOnly}, {ordered}, etc.

Most of the attributes describe the data associated with the objects in the class, but the associations between object classes also may appear as attributes or properties. In UML class diagrams, attributes and associations represent alternative ways to represent properties. An attribute in the Rental object class with the content of “lines: RentalLine [*]” shows an attribute with the name of “lines” that specifies the association between the Rental object class and the RentalLine object class, for example, “each rental object may be associated with many rental-line objects.” Similarly, the attribute “Date” in Rental might be represented as an association between the Rental object class and the Date object class.

As noted previously, all object classes are assumed to contain the CRUD operations of create, update, retrieve, and delete. These operations are not shown in the operations area of the object class. The Customer object class contains only the CRUD operations, so nothing appears in the operations area of the rectangle for the Customer object class. The Rental object class shows behavior in addition to the CRUD operations. A rental object must obtain from each associated rental-line object the data on the cost, add the costs to obtain a subtotal, and calculate and add the sales tax to get a total cost for the rental object. This behavior appears in Figure 5.12 as an operation called “calculate total” in the Rental object class. In similar fashion, the RentalLine object class might have operations to calculate the associated cost, perhaps the number of days the video is rented times the cost per day, and also an operation to calculate the overdue charge when a video is returned after the due date. The Video object class may contain other operations, but the other operations are not available to the rental system.

UML uses the term **features** to refer to both properties and operations of an object class. UML specifies a broad range of information that the analyst may include in a class diagram. Conceptual-level diagrams often contain mostly information about the organizational functions and roles. Physical-level class diagrams can contain all of the information needed to generate code. The physical version of the GB Video class diagram contains additional objects and much more detail than shown in Figure 5.12.

FIGURE 5.13
Sequence
Diagram for
the “calculate-
Total” Oper-
ation in Rental



Sequence Diagrams

A **sequence diagram** shows how objects behave for a specific scenario. The scenario may describe a use case or an operation in an object. A sequence diagram is an example of a dynamic, behavioral UML diagram, one that shows behavior over time. The diagram in Figure 5.13 shows a sequence diagram for the scenario of the “calculateTotal” operation of the Rental object class.

The rectangles labeled *aRental* and *aRentalLine* are objects—instances of their object classes. The arrow at the top left labeled *calculateTotal* shows the “found” message that initiates the sequence. The sequence starts with the activation of the *aRental* object shown by the vertical rectangle under the *aRental* object. The *aRental* object sends a *message*, **getCost*, to get the costs from each of the associated *aRentalLine* objects. The * at the beginning of **getCost* indicates iteration; the *aRental* object may need to obtain cost from one or more *aRentalLine* objects. The *aRentalLine* objects associated with the *aRental* object activate as shown by the box and return the costs. The number of costs returned is determined by the number of rental lines in the rental. Often the *return* arrow is not shown since the message to get a cost implies that the *aRentalLine* object will return the cost. Any box on the *lifeline* for the object represents the activation of the object. In the example, the *aRentalLine* object activates only once, but it might activate a number of times in a more complex scenario.

When the *aRental* object receives the costs, it self-calls or calls on itself to calculate a subtotal. After it completes the subtotal, the *aRental* object calls itself again to calculate the tax and add the tax to the subtotal. The sequence now has

calculated the total and is finished. The arrow at the left shows that moving down in the diagram represents the progression of time.

Advantages of Object-Oriented Design

Object-oriented design (OOD) represents an alternative way to define and implement the structure of a system. Many new systems are implemented in such object-oriented programming languages as C++ and Java. When object-oriented languages are used, the charts and other specifications in UML offer a consistent way to define the system at both conceptual and physical levels. The object representation of a system in a class diagram provides an obvious way to divide up the design tasks by assigning a person or team of people the responsibility to work on the design of an object or group of objects. The class diagram, once it exists, allows the teams to perform the detail design work largely independently of each other. The encapsulation of data and operations in objects with messages to invoke the interactions between objects also can simplify the maintenance and modification of an object-designed system. The analyst can make changes within an object with little if any effect on the rest of the system.

Object proponents cite reuse or **reusability** as a major advantage of an object design. For example, once a Customer object is designed, the same object can be used in many different systems that involve customers. The reuse also can extend beyond organizational boundaries. Libraries of standard objects exist and are growing. An organization that wants to design a new system can incorporate the standard objects that already exist into the new system. The reuse of a standard object may require changing some of the features of the object, but many of the features probably will transfer from organization to organization with little if any change. Object-oriented design is a step, and probably a significant step, but certainly not the last step on the never-ending quest for better ways to design systems.

Summary

Processes provide the engines for an information system. Processes accept data from sources outside the system, place data in and retrieve data from data stores, modify data, transfer data between processes, and send data to users outside the system. In short, processes accomplish all of the work or activity that occurs in a system.

Process models provide abstract representations of real systems that capture some but not all of the structure of the processes in the system in a form that people can understand. Most process models use graphical representations. Some process modeling tools that evolved from the structured approach to systems analysis include:

- Data flow diagrams (DFDs)
- Process hierarchy charts (PHCs)
- Input/process/output (IPO) charts

Data flow diagrams provide the most detailed representation of processes and their interactions. Process hierarchy charts, also called function hierarchy diagrams (FHDs), show how processes and subprocesses relate to each other. Input/process/output charts represent in table format the relationships between data coming into and going out of the system with a text description of the activities in each process.

Rules for preparing DFDs include:

- *Process dominance. Every data flow must come from a process or go to a process or both.*
- *Process completeness. Every process must have one or more data flows into the process and one or more data flows out of the process.*
- *Data store completeness. Every data store must have one or more data flows into the store and one or more data flows out of the store.*
- *Labels. Every component on a DFD must have a unique label or descriptor. Data flow labels consist of a word or phrase that describes the data in the flow. Externals have a noun label that describes the source or destination. Data stores have a noun label. Processes are labeled with a numeric identifier: 1, 2.0, 5.1, 8.1.1, and so on. In addition, processes have a descriptive phrase that varies depending on the type of the process.*
- *Process data sufficiency. The data flowing into a process must be sufficient to allow the process to (a) perform its tasks and (b) generate the data flowing out of the process.*
- *Data store sufficiency. The data placed or stored into a data store must be sufficient to populate any data flows out from the store.*
- *Context-level DFDs. All of the processes in the system are represented by a single process box with the system name as the description in the box. The outline of the box represents the boundary of the system. Each and every external in the system must appear on the diagram. Each and every data flow in the system to or from an external must appear on the diagram. Data stores, as such, must not appear on the diagram; by definition, data stores are internal to the system. Data stores created or used by other systems may appear on the diagram as externals.*
- *Decomposition. An explosion must contain at least two subprocesses.*
- *Consistency. Every data flow and only data flows into and out of the original process can cross the boundary box of the explosion and connect to one of the subprocesses. Each of these flows must have exactly the same name on both diagrams.*
- *Labeling of processes in explosions. In an explosion, compound processes receive a noun and an adjective description that ends with the word Process. BFMs receive a text phrase that begins with an active, present-tense verb and describes the action performed by the BFM. All processes receive a numeric label. The label for processes in an explosion is the label of the original process followed by a "." and an integer.*

The analyst can prepare a *hierarchy* of DFDs. The *context-level DFD* provides a good overview of the interaction between the system and the external environment, but it does not provide much detail about what goes on inside the system.

The act of providing more detail is called “*exploding* the process.” Exploding a process consists of the following steps:

- The analyst divides or decomposes the original process into subprocesses. The new subprocesses are placed on the explosion DFD.
- All of the data flows into or from the original process are connected to one of the subprocesses on the DFD.
- Data stores used by the subprocesses are placed on the DFD.
- Data flows are added to reflect retrieval from the data stores. Data storage flows are added and must satisfy the data store sufficiency rule.
- Subprocess to subprocess data flows are added as needed to achieve process data sufficiency.

The *process hierarchy chart (PHC)* or function hierarchy diagram is a process modeling tool that graphically displays the hierarchical relationships between processes in a compact format. The design process for a PHC follows the design for hierarchical DFDs. The *IPO (input/process/output) chart* is a process model used to study a system by identifying the input and output data flows. IPO charts consist of a table with three columns.

1. The left column contains the external data that flows into the system.
2. The center column contains descriptions for every one of the BFM processes in the system.
3. The right column contains the external data flows from the system.

Object-oriented design represents an alternative way to define and implement the structure of a system. Many new systems are implemented in such object-oriented programming languages as C++ and Java. The object representation provides a way to divide up the design tasks by assigning a person or team of people responsibility to work on the design of an object or group of objects. Object design also can simplify maintenance and modification. Object proponents cite reuse or reusability of objects as a major advantage. Object-oriented design is a step, and probably a significant step, but certainly not the last step on the never-ending quest for better ways to design systems.

With object-oriented design, programs and systems are structured not around data, processes, or events, but around objects, things that exist and play a significant role in the actual physical environment of the system. Objects contain within them the *data and operations* (processes) required to carry out their desired behavior. An object is an entity on steroids: It represents all the data for the instances and also all the behaviors that apply to the object class. Objects interact with each other by sending and receiving *messages*: “Here are some data and/or a task for you,” “Send me the data about x,” and so forth. Procedures or operations and data in an object are *encapsulated* or hidden.

The *Unified Modeling Language (UML)* provides a standard structure for object-oriented design and development. Three of the more commonly used UML charts are: (1) use case diagrams, (2) class diagrams, and (3) sequence diagrams. Use case diagrams give the analyst a way to display the interaction of system

processes with each other and the external environment. The use case diagram looks like a process model of the system and resembles the first explosion DFD. A class diagram shows the static structure of the system and represents the key or central diagram in an object design. The class diagram describes the objects in the system, identifies the data and operations for each object, and shows the relationships between objects. A sequence diagram shows how objects interact over time to carry out a scenario.

Key Terms

actor, 179	encapsulate, 178	process data sufficiency, 164
association, 181	explode, 165	process dominance, 162
attribute, 181	external, 159	process hierarchy chart (PHC), 157
basic function module (BFM), 170	feature of an object, 182	process model, 156
class diagram, 181	hierarchical DFDs, 165	property, 182
compound process, 170	IPO (input/process/ output) chart, 174	reusability, 184
context-level DFD, 165	label, 163	sequence diagram, 183
data flow, 158	message, 178	sink, 159
data flow diagram (DFD), 157	object, 177	source, 159
data store, 158	object class, 177	split data flow, 164
data store completeness, 163	object-oriented design (OOD), 184	Unified Modeling Language (UML), 156
data store sufficiency, 164	operations, 177	use case, 179
decompose, 162	process, 159	use case diagram, 178
elementary process (EP), 170	process completeness, 163	

Review Questions

1. For the data flow diagram (DFD), answer the following questions.
 - a. What is an elementary process?
 - b. Why should metadata for an elementary process normally be at the code level?
 - c. How does a DFD capture process logic?
 - d. What is common between a DFD and an ERD?
2. For a context diagram in a DFD, answer the following questions.
 - a. What is shown on a context diagram?
 - b. What is the connection between a context diagram and the first-level explosion?
3. Answer the following questions about rules for data flows.
 - a. What are the contents of a data flow?
 - b. What kind of DFD objects can be connected directly with a data flow?
 - c. What are the rules for data flow labels?
4. For explosions in a DFD, answer the following:
 - a. What kind of artifact can be exploded on a DFD?
 - b. What are the naming conventions that connect an explosion to the higher level diagram?
 - c. What must an explosion have in common with the higher level diagram?

5. Answer the following questions about an IPO chart.
 - a. Which processes are on both models?
 - b. Why does an IPO chart not include entities?
 - c. Which flows should appear on both IPO and DFDs?
6. Answer the following questions about a process hierarchy chart.
 - a. What is a process hierarchy chart intended to show?
 - b. What is the difference between a process hierarchy chart and an IPO chart?
 - c. What is common between a process hierarchy chart and a DFD?
7. Define the following object oriented terms:
 - a. Use case diagram
 - b. Class diagram
 - c. Sequence diagram
8. How do objects handle the following parts of a system design?
 - a. Process analysis
 - b. Sequence of code execution
 - c. Data modeling
9. For an object model, answer the following:
 - a. What is the difference between an object and an object class?
 - b. What are CRUD operations?
 - c. What is encapsulation?
 - d. What is a message?
10. Under what circumstances would you choose the following model approaches?
 - a. DFD model
 - b. Object model
 - c. IPO model

Critical Thinking Exercises

Individual Exercises

1. The director of a bowling tournament needs a database to connect PLAYERS with MATCHES. The database should contain data on PLAYER-NAME, PLAYER-PHONE, GAME-TIME, LANE-NUMBER, and SCORE for each player.

Players sign up with a team. Players may play for several teams, but for only one team in a given league. Once enough teams have been enrolled to form a team, the league sets up a schedule that determines who plays who each week.

As teams play each other the league keeps track of team victories, team total pins, and individual game averages. Each match consists of three head-to-head games. The team score is calculated by adding one point for the winner of each game plus one point for the high total points plus handicap. For scoring, each player receives his or her total number of pins *plus* the player's handicap.

This league recalculates handicaps after the season not on an ongoing basis using an 80 percent of 200 basis. The handicap is calculated on the best six games for each player. The handicap is given by subtracting the average of these six games from 200 and multiplying that difference by 80 percent. If the six-game average is higher than 200, there is no handicap. No handicap can be larger than 20 pins.

At the end of the season, the league calculates the team winners and the individual scoring winners and then awards trophies.

Draw a data flow diagram for this system. Make sure that the files in your model match the entities you found in the data model from Chapter 4.

2. A car rental company classifies customers as business, leisure, and urban. The company wants to keep name, address, and home phone number for all rental customers. Specifically for business customers, the company wants to keep the business name and business phone numbers; for leisure customers the company needs destination(s); and for urban customers the company needs the insurance company and expected numbers of days needed. When customers arrive at an office they are assigned a vehicle (VIN, make, color, license) and a rate depending on how they reserved the vehicle. When they return the vehicle, the agency records days rented and number of miles. Each office owns the vehicles it rents. The database keeps the information on location, manager, and owned vehicles.

When a customer calls in for a reservation, the clerk enters the basic customer information into the system. The system checks the car inventory to be sure that a vehicle will be available when it is needed and returns a confirmation with confirmation number and other necessary information.

When a customer arrives at the rental office, the clerk checks the system to retrieve the reservation information, gets credit card and insurance data from the customer, and assigns a car to the customer for use. If the customer has not reserved a car ahead of time the clerk will go through both the reservation and delivery process at the counter.

Once the customer returns the car, the clerk checks it for damage and enters any damage or cleaning charges. The system then calculates the bill based on time and mileage according to the rate package that the customer signed up for and prints a receipt. Once the customer signs the receipt, the system bills her/his credit card for the correct amount.

Draw a data flow diagram for this. Make sure that the files in your model match the entities you found in the data model from Chapter 4.

3. For the process of running a babysitting service at the University (see Group Exercise 3 in Chapter 3), do the following:
 - a. Prepare an IPO, a process hierarchy chart, and a DFD.
 - b. Construct a use case and a class diagram.
 - c. Prepare a sequence diagram for one of the operations in the class diagram.
4. Draw an IPO, a process hierarchy chart, and a DFD for setting up and managing a professional club for the IT students in your program of study.

Group Exercises

1. Use Group Exercise 4 in Chapter 3 to complete the following exercises.
 - a. Draw an IPO, a process hierarchy chart, and a DFD for the system.
 - b. Construct a use case and a class diagram.
2. Sooner Events is a service organization that runs business retreats at their lakeside camping facilities. The service wants an interactive system that will allow their customers to manage their own schedules.

The service gave the following description of what they want. Customers can select camps on their own. The system should display the camp topics and times that Sooner supports. Customers can schedule camps, guarantee attendance, and select special activities that go with them. The customer will provide a list of attendees. Individual attendees will register with Sooner, guarantee their bill, and select any special needs or meals they might need during the camp time.

Behind the scenes the system should support the process of scheduling event staff to provide services. Planners working for Sooner will interact with the system to select

and contact contract employees who deliver the seminars, order refreshments and amenities, and schedule special events. In addition, the system should allow the lodge manager to schedule blocks of rooms in the lodge and to inform the kitchen of any special needs.

Draw a use case and class diagram for this system.

References

- Fowler, Martin. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, 3rd ed. Reading, MA: Addison-Wesley, 2003.
- Gane, Chris; and Trish Sarson. *Structured Systems Analysis and Design Tools and Techniques*. Upper Saddle River, NJ: Prentice Hall, 1979.
- Hoffer, Jeffrey A.; Joey F. George; and Joseph S. Valacich. *Modern Systems Analysis and Design*, 4th ed. Upper Saddle River, NJ: Prentice Hall, 2005.
- Rumbaugh, James; Ivar Jacobson; and Grady Booch. *The Unified Modeling Language Reference Manual*. Reading, MA: Addison-Wesley, 1999.
- Schneider, G.; and J. P. Winters. *Applying Use Cases: A Practical Guide*, 2nd ed. Reading, MA: Addison Wesley, 2001.
- Yourdon, Edward; and Larry Constantine. *Structured Design: Fundamentals of a Discipline of Computer Program and Design*. Englewood Cliffs, NJ: Yourdon Press, 1986.
- Whitten, Jeffrey L.; Lonnie D. Bentley; and Kevin C. Dittman. *Systems Analysis and Design Methods*. New York: McGraw-Hill/Irwin, 2005.

Project Definition

Projects are undertaken on behalf of a client and a sponsor. Normally, the client requests a project to solve a problem. The client communicates to the team such information as goals, constraints, and features that the client wants to guide the team to a satisfactory solution. In initial communications, the client may neglect to mention some important information and may become aware of additional constraints and desired features as the project proceeds. Project definition represents a carefully structured effort by the team to discover and document the client's requirements and to help the client clarify and refine the goals, features, and constraints for the project. Project definition occurs at the beginning of the project, but as noted in the discussion of the spiral model, the team may need to revisit the project definition at other points during the project as both the client and the team learn more about the issues.

Chapter 6 explores the concepts and issues of determining goals, features, and constraints for solutions to the client's problem. A project is successful only when the project contributes to the goals and values of the organization and the client. Elegant and technically advanced solutions may appeal to the team, but may or may not contribute to the client's goals or values. Strategic alignment addresses how to focus the project to contribute to such strategic values or goals of the organization as profits, quality, service, sales, and customer satisfaction. The team conducts the strategic analysis at the beginning of the project and uses the insight gained from the analysis to make decisions throughout the project. The term *performance-oriented design* embodies the idea that the team makes all project decisions in the context of providing performance or value to the client.

Sometimes the client provides all the information that the team needs without prompting. Often, though, the team will want to ask questions, make observations, and collect additional information. The team needs information to define the desired features for the solution, including the functionality and/or performance the client wants that does not exist in the current situation. The team also needs for the client to define constraints for the project, for example, time and budget limits. The team explores acceptable solution options with the client. The client may be willing to do one or more of the following: purchase a solution, build a solution, or continue with the current situation. When the team has

Chapter Six

Understanding the Client's Problem and Organization

Chapter outline

Introduction

Strategic Alignment

The Organization Case

The Organization

The Organization's Goals and Objectives

Project Contribution

Project Success

Determining Alignment

Understanding the Organization

Identifying Objectives and Goals

Identifying Project Contribution

Project Success

The Project Definition Report

Project Statement

Strategic Alignment

The Organization

The Organization's Goals and Objectives

Project Contribution

Project Success Criteria

Proposed System Features

Constraints

Constraint Types

Solution Options

Scope

Examples of Project Definition Materials

Working with the Client

Professional Behavior

Prepare for a Visit

Make a Visit

Information Collection Approaches

Interviews

Group Interviews

Documents

Observation

Surveys and Sampling

Summary

Key Terms

Review Questions

Critical Thinking Exercises

Individual Exercises

Group Exercises

References

INTRODUCTION

As previously discussed, the process of generating an information systems solution consists of four stages: (1) project and team organization; (2) project definition; (3) proposed system; and (4) system delivery. During the first stage, the client proposes a project. Once the project is approved, the team forms, organizes, and prepares the initial plan. Phases, of course, can overlap and interact. The team may and probably should work on team and project organization throughout the project. However, once the initial team and project organization work is done, the team can move on to the next stage, **project definition**.

The basic goal of the project definition stage, the first stage during which the team interacts with the client, is to acquire and analyze information to answer the following questions:

1. What does the organization value? What are the goals, objectives, and the key measures of organizational performance for the organization?
2. Can the proposed solution add value to the organization? Can it enhance or contribute to the performance measures of the organization?
3. Who is the client—the person or group with the authority to sponsor or define the project, and what does the client value?
4. What features and constraints for the proposed information system solution does the client want?
5. What can the team learn from the current situation for the design of the proposed system?
6. Can the proposed solution solve the client's problem? Can it eliminate or reduce the problems and/or achieve the client's objectives?
7. Is the project scope reasonable?

The first three questions in this list relate to the issue of strategic alignment. From the beginning of the project, team members want to understand how the project fits into such strategic values or goals of the organization as profits, quality, service, sales, customer satisfaction, and similar concerns. To address the alignment question, the team must identify what the organization and sponsor value, sometimes a difficult task. When the team understands the organization's values, the team can relate better with the clients and can focus the project work on achieving outcomes that contribute to the organization's values. At each point in the project, the team should follow the path that looks most likely to result in a system that will add value to the organization.

The team refines the project plan to identify the activities to perform during project definition. Some typical activities include the following:

- Establish contact with the client and arrange for one or more visits.
- Plan, prepare for, and make the visits with the client to collect information.
- Assemble the materials on strategic alignment, features, constraints, and alternatives materials.

- Assemble the data, process, infrastructure, problems, retention, and change materials for the current situation or operation.
- Prepare the draft version of the project definition report.
- Conduct a project definition presentation with the client.
- Complete the final project definition report.

The team carries out information collection, analysis, and modeling related to the problems, goals, requirements, constraints, and alternatives for the project. **Information collection** represents a primary activity during the project definition stage of the project. Team members should prepare a plan for the information they wish to collect and the approaches that they will use to collect it. Normally information collection requires a number of visits to the client's location. After each visit and subsequent analysis and modeling, the team may identify additional information to collect. The project definition presentation with the client provides an excellent opportunity to identify and fill in missing or incorrect information.

One obvious way to begin finding answers to all of the project definition questions is to ask the client. The team should ask the client and note carefully the answers. In some uncomplicated projects, the client may say something like "Here is the definition for my project: The organizational values, what I want, when I want it, and how much I am willing to spend." In most cases, the process involves more complexity. Different people—senior managers, functional managers, and users—may have different desires and perceptions of problems and requirements. A nonuser client may not understand fully the current operation and may not know all the problems. The client may not know what is possible, what are the reasonable solutions, and more importantly what are the trade-offs. In other words, what are the costs, benefits, and risks of the alternatives? The team bears the responsibility to help the client clearly understand all of the relevant issues.

STRATEGIC ALIGNMENT

Organizations sometimes say they want better information systems, but actually organizations want better performance with respect to overall organizational values or goals: higher stock prices, more profit, higher sales, lower costs, more satisfied customers, and so on. The client believes or hopes that a proposed information system will bring value to the organization by improving one or more important measures of performance. In contrast, many information technology (IT) teams, and even some IT managers, operate with little thought about the relationship between information systems and organizational values. IT people tend to focus on requirements, analysis, code, problems with existing systems, schedules, and costs.

To succeed, an IT team needs to understand and think like the client. In well-run organizations, managers at all levels and in all areas of the organization are constantly reminded to focus on strategic goals or on critical organizational values. One of the recurring themes in discussions of the most critical information

technology issues is the alignment of IT activities with the organization. Alignment of IT activities means focusing IT activities to support organizational values.

A project team should understand and support organizational values for the following interrelated reasons:

- To gain organizational *acceptance* and *credibility* for the team with the rest of the organization. Many modern organizations operate as a network of teams. Senior managers think of themselves as heads of a team of workers who identify with each other and work together for a common goal. Critical factors in the success of any project include strong managerial and organizational support. Unless the IT people can “talk the talk” of the rest of the organization, they will remain peripheral to and less supported by the larger organizational team.
- To obtain and retain *financial sponsorship*. IT projects often receive funding not from the IT budget but from the budget of one of the core operating units of the organization. For example, a marketing division might fund a Web-based product sales system or a vice president of manufacturing might pay for a supply chain management system. To succeed, a project must obtain a sponsor who will pay for the project and the project must retain the support of the sponsor even if and when predetermined conditions change, as often they will. The team should demonstrate that it understands the organizational values and that the project will contribute enough to organizational values to warrant the support of the sponsor.
- To justify the **recommended solution**. Managers are usually cost- and value-oriented. They resist spending money without understanding the value they will receive in return. Accepting an IT recommendation incurs costs, often large costs, and non-IT managers seldom understand fully either the costs or the expected value. The managers must rely on the IT professionals to supply sound organizational judgments. IT teams must justify projects and recommendations on organizational values not on the merits of the technology or the elegance of the solution. Values can encompass a range of things. A police department might value a reduced crime rate while an art museum may measure success in increased attendance.
- To make good ongoing project *decisions*. The project team makes a continuing sequence of decisions. How much functionality, performance, security, and so forth is enough? How important is an audit trail? And so on. The team should make decisions in the context of the **impact of a feature** on organizational values. Some decisions may involve trade-offs between system features and project resources, and the clients may not understand the consequences of the choices. When the consequences are significant, the team must present the choices to the clients and sponsors and explain the potential impact on organizational values.

The Organization Case

The team collects information and prepares an organizational case to establish and maintain the alignment of the project with organizational values. The organization

case for **strategic alignment** rests on three components: the organization, goals and objectives, and project contribution. These dimensions provide a way to align any project, IT or other, with the goals of the organization.

The Organization

As noted, the team needs to understand the organization to gain acceptance and credibility. Senior managers seldom talk about DFDs, foreign keys, or Java. Managers talk about customers, employees, products, plants, equipment, operations, sales, service, billing, profits, and stock prices. Some of the organization attributes that could hold importance for the project include mission, vision, culture, organization charts, authority and responsibility, available resources, legal and regulatory issues, security, glossary of key terms, and a description of current operations. The project may result in changes to the organization. Only a few aspects of the organization will produce a major impact on the project and those few are the critical ones to understand and document. For additional discussion of organizational style and its impact on information systems, see Kendall, 2002.

The Organization's Goals and Objectives

The central issue of alignment is to assure that project outcomes support the goals and objectives of the organization. A **goal** is an expression of a commitment to a state that the organization wants to achieve, for example, to become more profitable is a goal, albeit an elusive one, for every airline. Typical high-level goals refer to such areas as profits, cost-control, revenue enhancement, sales, quality, service, and customer satisfaction. An **objective** is a concrete, measurable action that supports that goal, for example, to increase airline revenues by increasing aircraft utilization with faster turnaround and better scheduling. When possible, objectives are stated in terms of such outcome measures as profits, revenue, costs, turnaround time, sales, inventory reduction, and more. These outcomes provide the basis for evaluating success in the organization and for the IT system.

Some organizations conduct periodic strategic planning exercises in which they review past operations, evaluate challenges and opportunities, and identify goals and objectives for the next time period. Other organizations identify goals and objectives in less formal ways. The goals and objectives that come out of these high-level meetings drive the prioritization process for investments, including investments in IT. Identifying the strategic framework of goals and objectives that relate to the project forms a critical step in determining value for a project.

Project Contribution

The final and critical step to strategic alignment consists of identifying the contribution of the project to the strategic objectives of the organization. At this point, the team understands the organization and has selected some goals that relate to the project. The team either finds or formulates objectives that (1) lead toward the selected goals and (2) are facilitated by the project. Often, the objectives target improving a part of the organization that does not function as well as desired. In this event, the team identifies the problems and constructs

features for the proposed system that will improve performance with respect to objectives. This process is known as **performance-oriented design**.

The team should identify specific, tangible **performance measures** that the proposed system will impact. The performance measures must relate directly to the objectives. For example, if the team selected an objective of reducing the cost of selling the product, then the appropriate performance measure is sales cost. Some objectives are more difficult to measure; for example, to improve customer satisfaction by giving the customer access to an electronic ordering system. Accounting standards, laws, and regulations do not define how to measure customer satisfaction. Perhaps customer satisfaction can be measured by surveys or by an increase in orders, but measuring customer satisfaction is more difficult than measuring costs.

Project Success

The sponsor and/or lead client formally or informally will hold in mind a set of criteria for a successful project. The **project success** measures may involve one or more organizational performance measures, for example, the project is a success if sales costs are reduced by 10 percent. Often measures of success also involve project parameters, such as to complete the project on time and on budget. Student projects involve an additional success measure not directly related to the client, such as to meet the grading criteria for the instructor. While teams may pursue many goals, most teams want to achieve a “successful” project.

Determining Alignment

The team normally performs strategic alignment only once at the beginning of the project. The team conducts the strategic analysis early in the project because it provides the context or a frame of reference for addressing the critical issues in the following stages. The team uses the strategic alignment analysis throughout the project to identify features for inclusion in the system, to set priority of activities in the project plan, and to evaluate the merit of system alternatives. The mission, objectives and goals, and key performance measures for the organization normally will remain constant over the life of a project.

Understanding the Organization

The team starts the alignment process by making sure that it understands the organization’s values. Teams that already work in an organization will start with some organizational understanding. Teams from outside the organization must start at the beginning to build understanding. The process of understanding an organization usually starts with an understanding of the mission and vision of the organization and sponsoring unit. The **mission statement** tells the current major focus of the organization. Mission statements, when they exist, often appear in such documents as annual reports and/or on the Web. These statements should provide insight on what the organization thinks is important or, at least, what the organization wants other people to believe is important.

The **vision** of an organization is the understanding or picture of what the organization wants to be in the future. Sometimes vision statements appear in

documents; other times the vision is a general agreement among key members of the organization. The client may provide organization charts, insight into organizational culture, procedure or standards manuals, and other information on the organization. During project definition, the team will analyze current operations as another way to gain an understanding of the organization.

Organizational culture can impact the project. Organizations, like people, develop dogma, preferences, and opinions that become part of the decision structure of the managers in the organization. For example, some organizations purchase applications whenever possible while others prefer to build their own. Some like to make decisions fast using mostly experience and intuition with little if any formal analysis. Others demand extensive analysis and a comprehensive written review for even minor decisions. Tolerance for risk is an important part of culture. In organizations that are risk adverse, the team wants to pursue and recommend low-risk alternatives. Clients often provide insight into the organization's culture during discussions. For example, the client may say something like "We always do it this way" or "We never do that."

Identifying Objectives and Goals

A critical step in aligning a project to an organization is determining what the organization values most. Goals and objectives are more precise statements of organizational values related to the organization's perceived vision and mission. If the organization practices formal strategic planning, then the senior executives meet periodically to revise and review the objectives. Objectives usually translate into specific assignments for senior managers on which their performance is measured. Supporting critical organizational objectives is the highest priority of most potential sponsors.

Some organizations have a documented strategic plan with stated goals and objectives and may share the plan with the team. Organizations also may have a strategic IS plan that identifies and prioritizes IS projects and/or systems in support of the organization's strategic objectives. Clearly the team should collect any information on strategic plans that the client is willing to share. If large, multiple-page strategic plans exist, the team can extract the relevant parts for the project.

Other organizations, particularly smaller ones, may not prepare a strategic plan. In this event, the team can work with the client to obtain a mutually acceptable strategic framework. Asking a client to list goals and objectives may elicit little useful information. Clients often are preoccupied with the most current hot issues and may look blank when asked to list their goals and objectives. In many situations, team members will need to infer goals, objectives, and other values from their discussions with the clients. The team can develop their own list and ask the client to comment and expand on the list. The mission statement may provide the starting point for a discussion of goals and objectives with the client. Mission statements can indicate what an organization thinks are important values; however, the team should approach mission statements with some care. Mission statements often present public relations messages rather than a careful recitation of organizational goals, objectives, and values.

TABLE 6.1
Goals and
Related
Objectives for
a Video
Rental Store

Goals	Related Objectives
1. Increase profitability	1.1. Reduce the labor cost of renting and returning videos 1.2. Reduce the cost of nonmembers renting and not returning videos
2. Increase customer satisfaction	2.1. Reduce the average checkout time for a rental

The team may end up with a list of goals and objectives, only some of which hold any relevance to alignment issues for any particular project. Some of the objectives may not match up with the project in any way. The team should focus on the important objectives that the project can support. For example, a reservation system for an airline may support the objectives of reducing operations cost and decreasing oversold flights but probably brings little value to an objective of improving refueling safety. In addition to identifying the specific objectives that relate to the project, the team also wants to identify the objectives valued by a sponsor or potential sponsor of the project. The most important objectives for a marketing sponsor probably will differ from those valued by the vice president of operations. In summary, a good project team focuses on the several objectives that (1) the project sponsor values and (2) the project can support. A clear statement of these objectives will appear in the project definition report. The team may wish to record the goals and related objectives in a table similar to the one shown in Table 6.1 for a video rental store.

Identifying Project Contribution

Once a team has a clear idea of the relevant objectives, the team can move on to identify how the project can help achieve the objectives. Most organizations produce something: a product, a service, a capability, or an idea. The client's problem often refers to a part of the organizational process that performs at a lower than desired level. Often, the objectives target improving a part of the process. The team's task is to understand the production process for the product or service, to identify the parts that do not work correctly, to relate the problems to objectives, and to identify how the features of the proposed system can help to achieve the objectives. The team identifies the features of the proposed system as part of project definition.

A possible approach to identifying how the system can contribute contains the following steps:

1. Obtain from the Goals and Related Objectives table the list of objectives relevant to the proposed system that the team and/or client identified.
2. Obtain the list of features for the proposed system. A discussion of obtaining features for the proposed system appears in the Project Definition Report section of this chapter. The analysis at this point is conceptual—as free as possible of technologies. Avoid features that involve or imply a specific technology unless the technology is a basic part of the project statement from the client. If the client wants a Web site, the Web site part of the technology is given, but such more specific technologies as Java should not appear as features.

3. For each objective, analyze and, if possible, quantify, the impact of each of the features for the proposed system on the objective. In order to measure impact, a goal must have a performance measure. In a well-designed strategic plan, objectives are stated in terms of measurable objectives. If the organization currently has no such measures, the team will need to work with the client to generate performance measures. Most people think first of cost savings, but other measures may hold more importance in some situations. The team and client must work together to determine appropriate measures and the project's potential for improving the measures.
4. Each new feature should impact at least one of the objectives.
 - a. If a feature does not match up with an objective, review the objectives to see if one is missing. Remember that each objective must contribute to one of the organizational goals. If no objective matches the feature, consider deleting the feature. Discuss the situation with the client before acting.
 - b. If an objective exists with no features that impact it, then review possible features for the new system to try to find a feature that will impact the objective. If no reasonable features appear to impact the objective, drop the objective from the analysis.
5. Try to generate an estimate of the impact on each objective. Sometimes the impact on the objective comes from a single feature; other times the impact may come from a combination of features or from the system as a whole. If the objective is to increase sales, then obtain an estimate of the impact on sales—perhaps, sales will increase by at least 5 percent. The client is the best source of such estimates.

The process reinforces the concepts of performance-oriented design. The key features of the proposed system relate to improved performance measures for objectives of the organization. The team can place this information in a table such as the one shown in Table 6.2.

When the table is complete, the team may have a large number of feature/objective impacts. In reports, the team should discuss and stress the most important or higher impact ones. The team even may wish to eliminate the less important objective/feature impacts in the project definition materials prepared for the client.

TABLE 6.2
Feature
Impact Table
for Video
Rental

Objective	Measure	Feature	Impact
1.1. Reduce the labor cost of renting and returning videos	Labor cost	Automate process	\$50,000 per year
1.2. Reduce the cost of nonmembers renting and not returning videos	Cost of unrecoverable videos	Access rental process only from a confirmed member	\$5,000 per year
2.1. Reduce the average checkout time for a rental	Checkout time	Automate process	Decrease from an average of 4.5 to 1.5 minutes

Project Success

Sponsors and, with student projects, instructors determine the success of a project. The team may believe that the project was a resounding success while the client is disappointed. Alternatively, the team may regret the lack of more elegance in the solution while the client believes that the solution is wonderful. The folklore joke about surgeons, “The operation was a success, but the patient died,” sums up the dilemma. The client wants a healthy patient, that is, an outcome that solves his/her problem, not a project with a dead patient that the team thinks was a success.

The team discusses the project success criteria with the sponsor or key client. The team should ask in detail about the deliverables the client wants, the budget and time constraints, any organizational performance objectives that the client expects, and the trade-offs important to the client. For example, the client may say, “I absolutely must implement the solution by a certain date. Do the best you can to include all the features in the specs. But if you cannot do everything, drop feature A first, then B if needed, but above all make sure that you meet the completion date.” A good definition of *success criteria* goes in the statement of work.

The instructor often expresses the educational or learning success criteria for a project in the course syllabus, assignments, grading sheets and instructions, and standards manual. The instructor’s learning success criteria may differ significantly from the ones expressed by the client. The learning objectives include acquiring the knowledge appropriate to carry out larger and more complex projects than the ones typically used for class.

THE PROJECT DEFINITION REPORT

The project definition report sets forth the team’s understanding of the client’s problems and requirements in an organizational context. At this point the team focuses on understanding the client and helping the client to express fully the client’s views and requirements. The team will have an opportunity during proposed system design to innovate or create solutions for the client’s problems.

The project definition report is “client-centric” in that it reflects only the client’s views and desires. Many, if not most, projects take longer and/or cost more and/or produce less than expected because the team does not identify fully and correctly the client’s desires and requirements. The difficulties may result from the client’s inability to express requirements or from the team’s lack of focus and effort to understand the client or from a combination of both. The report normally contains the following materials:

- Title Page
- Table of Contents
- Executive Summary
- Introduction

- Project Definition
 - Introduction
 - Project Statement
 - Strategic Alignment
 - Functions and Features
 - Constraints
 - Current Situation (see Chapter 7)
- Proposed System (stub)
- System Delivery (stub)
- Appendix A. Statement of Work (see Chapter 3)
- Appendix B. Documents supplied by the client
- Other Appendixes as needed

The team may select from several options for the project definition report. The option shown above uses the format in Chapter 3 for the final report with the Proposed System and System Delivery sections stubbed. A stubbed section consists of a section heading with no content or the phrase “To be prepared later.” A second option is to omit the Proposed System and System Delivery section headings. The team manager may specify the format that he or she wants.

The following materials discuss the project statement, strategic alignment, functions, and features for the proposed system, as well as the constraints and an exploration of alternatives in more detail. The introductory materials (executive summary, table of contents and introduction, and the statement of work) are covered in Chapter 3. A discussion of learning from the current situation appears in Chapter 7.

Much of the project definition material appears in narrative model or text representation. Clients find text descriptions and tables easy to understand. In addition, standard graphical models to represent such things as a project statement or strategic alignment do not exist. In the current situation materials, both narrative and graphical models are used because standard graphical data models (ERDs) and process models (DFDs) do exist. Creating both narrative and graphical models provides a powerful tool with which to check for consistency and completeness.

Project Statement

The **project statement** defines the project in a short paragraph. Subsequent sections on features, constraints, and the current situation expand and clarify the basic problem statement. A project statement for the GB Video rental system might read

Design and acquire a new computer-based system for the rental and return of videos because the current manual system maintains inadequate records and is too slow. The new system should address improved customer service and lower handling costs for each transaction.

The client normally specifies the project statement although the team may wish to restate or expand on it for clarity and completeness. Ideally, the project

statement should address only logical requirements and not specify specific solutions, software, or hardware. In practice, the client often specifies general or specific technologies, such as “I want a computer-based system” or “I want to go from paper records to images on optical storage.”

Strategic Alignment

As noted in the previous section, the team normally performs a strategic analysis of the organization only once, at the beginning of the project. The strategic analysis focuses on the strategy and values of the company or organization and not on the details of the information system. The team uses this analysis throughout the rest of the project to identify features for inclusion in the system, to set the priority of activities in the plan, and to evaluate the merit of system alternatives. The preceding section also describes the four components for the organization’s case for strategic alignment: (1) the organization; (2) goals and objectives; (3) project contribution; and (4) project success criteria. The team discusses each of these areas in the strategic alignment section of the project definition report.

The Organization

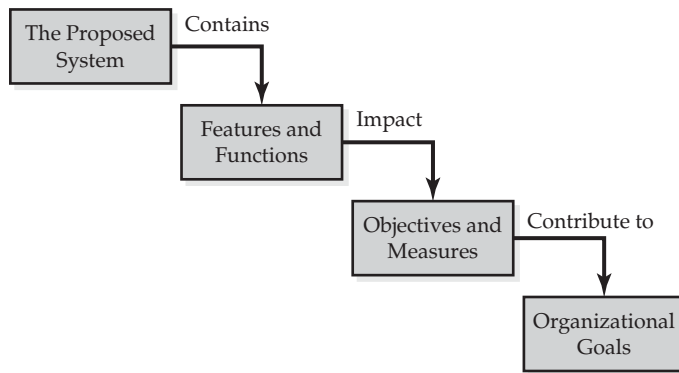
The **organization** section highlights the parts and aspects of the organization that relate closely to the project. Typical content for the section can include mission statement, vision, organization description, and organization charts. The team should include any formal mission statement if one exists, but augment general mission statements with a more focused vision statement about what the company actually wants to achieve. For example, if GB Video offers the general mission statement, “Serve the Customer,” the team can complement this statement with a vision: “Offer for rental the best available selection of videos in the market area at competitive prices and with better customer service than competitors.” The description of the organization identifies products, services, locations, and organizational structure including the relationship to a larger organizational unit if relevant. The team can include an organization chart or describe a simple organization in text.

The organizational material should maintain a focus on the project. For GB Video, the section describes the rental and return areas of the organization—the retail stores, clerks, and so on. A detailed description of the purchasing and personnel parts of the GB Video organization may offer less value for a rental system project.

The Organization’s Goals and Objectives

In the project definition report, the team identifies the goals for the organization and the objectives that may help the organization to achieve the goals. The team may wish to try to identify the full set of goals for the organization and to identify only those objectives that relate to the proposed system. Some of the objectives probably will not relate to the project, and some of the goals may not match any of the objectives that relate to the project. The team can retain the non-matching goals and objectives or can omit them. The client may want to see a

FIGURE 6.1
Proposed
System
Contribution
Model



full set of goals and objectives, including ones that do not relate directly to the project. Objectives that are supported by features of the proposed system warrant a central focus and full discussion. A table similar to the one in Table 6.1 can focus and clarify the discussion of goals and objectives. The table should serve as the basis for the discussion not as a substitute for a fuller discussion.

Project Contribution

A critical step to strategic alignment consists of identifying the contribution of the project to the strategic objectives of the organization. As described earlier in the chapter, the strategic alignment analysis follows a top-down model—start with the mission and vision of the organization, and work down to how the proposed system features contribute to objectives. The proposed system contribution section in the strategic alignment part of the project definition report is usually stated in the reverse order following the flow of Figure 6.1. The team notes that the proposed system contains features that impact the company's objectives and the associated performance measures and that the objectives support organizational goals.

The team provides the following information about project contribution:

- Features of the system and the objectives that each feature supports.
- Specific, tangible performance measures for each objective that the proposed system will impact.

Table 6.2 offers a convenient way to summarize and highlight the content of the discussion of objectives, features, and performance measures.

Project Success Criteria

The team provides a detailed discussion on the project success criteria as defined by the sponsor. The discussion expands on the summary information on project success criteria in the statement of work.

Proposed System Features

The solution process starts with and builds on what the client wants to accomplish, or on solving the client's problem. The team will refine and expand as

necessary the client's set of features during the proposed system phase to arrive at final specifications for the proposed system.

The client may identify several kinds of desired features for the proposed system including:

- System **functions** include the organizational functions that the new system should and should not include. Functions are the activities the organization performs to accomplish its mission. For example, GB Video might state that the proposed system should include functions for enrolling new members, renting videos, returning videos, and preparing reports and should not include functions for keeping accounting records, managing inventory, purchasing videos, billing customers, and sending out advertising.
- System **structure** encompasses, for example, automating the input, automating the processing, entering input data only once, storing the data in a computer database, having only one customer database for multiple users, and so forth. Information systems provide data and processes to support the performance of organizational functions.
- System **performance** covers features to eliminate errors, reduce the time to process a transaction, strengthen security, and so on. The performance of the information system can determine how well the organization carries out the functions for its mission.

The team works with the client to identify features for the proposed system. The team can begin by asking the client to identify the desired functions, structure, and performance. In a number of cases, the team will need to use the client's answers as a starting point, and then develop the specific features and present them to the client for approval. For example, the client may specify only some general goal, such as to automate the current manual system. With a general goal, the team will need to work with the client to develop a more specific and complete set of feature statements. When the client asks the team to automate the current system, the client may mean that the desired functions for the proposed system are the same as for the current system. However, often the client also wants changes in some functions and just assumes that the team will understand what is wanted. Miscommunication between client and team causes a large number of the problems that arise during system solution.

Often the goals as stated by the client relate to eliminating or mitigating problems with the current system. The team can gain significant insight from thinking about how to improve a current system. However, whatever the source of the features, the team also should relate system features to the organizational goals and measures identified in the strategic alignment.

Sometimes the client will propose incompatible or impossible features, for example, eliminating all the data errors, developing a system with maximum functionality at minimum cost, or developing an airline reservation system with a function that will predict exactly how many passengers will show up for each flight. No matter what system the team builds, some data errors probably will occur and some flights will end up over- or underbooked. And the system with the best or fullest set of functions probably will cost a lot more than the "minimum

cost” system. The team can work with the client to reach more meaningful statements, for example, “Eliminate 99 percent of the data errors in the current system or include as much functionality as possible within a budget limit.”

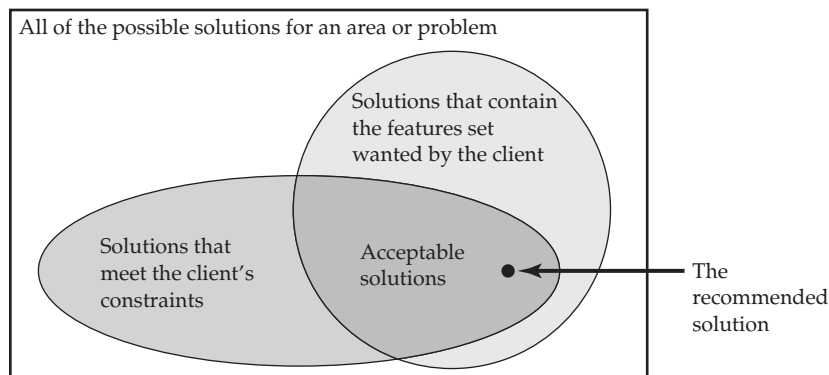
The team should place major attention on identifying the features that the client wants. The team can ask a lot of questions, talk with the relevant people, and collect copies of all of the relevant documents. However, despite the best efforts of the team, the client may and often does change desired function, structure, and performance as the project goes along.

Constraints

The team can devise many proposed solutions that contain the features desired by the client. Features together with **constraints** define the set of feasible solutions to a client problem as shown in Figure 6.2. A features set acceptable to the client identifies a subset of solutions from the universe of all solutions relevant to the problem area, for example, the area of video rental. The constraints identify a second subset of solutions that the client can afford or is willing to consider. The overlap, if any, of the features and constraints subsets defines the set of acceptable solutions—ones that provide the desired features and that the client is willing to consider. After further analysis, one of the acceptable solutions becomes the recommended solution.

The model shown in Figure 6.2 provides a good perspective for purchased solutions with the rectangle showing all the available package solutions for the area, the circle showing the ones with the desired features, and the ellipse showing the ones within the client’s constraints. Some of the solutions with the desired features may not meet the client’s constraints, perhaps, for example, they are too expensive. And some of the solutions that meet the client’s constraints may not contain the desired features. The recommended solution contains the desired features and exists within the constraints. For a build option, the number of possible solutions probably approaches infinity. The team normally focuses on a limited set that spans the space, for example, the solution with the best of everything, the minimum cost solution, the low-risk solution, and several in the middle solutions.

FIGURE 6.2
The Role of
Features and
Constraints in
Finding
Solutions



Constraint Types

The team, as with most issues, should ask questions about and explore possible constraints with the client. The team should come prepared with as much information as possible. The simple question, What are the constraints? may not elicit much of a response from the client. The well-prepared team can move from general, open-ended questions to much more specific questions, for example, What is the maximum amount you are willing to spend for the new system? Examples of possible constraints include the following:

- *Budget.* This is the maximum amount that the client wishes to spend in total and/or for various parts of the system solution effort.
- *Time.* This is when the client wants a particular deliverable. For student project teams, the client's time line may not match the class time line.
- *Function.* Is the client willing to change the way the process operates, to simplify the system, or to fit an existing package?
- *Organization.* Does the client wish to keep the current organization or is change possible?
- *Physical infrastructure.* Should the team plan to use the existing hardware, software, and communications or is change possible?
- *Procurement options.* Which options will the client consider: keep and refine the existing system, build a solution in-house, buy a package, outsource development or services?

Solution Options

The set of **procurement options** that is acceptable to a client represents an important and often difficult-to-define constraint. The team wants to spend time on only those solutions that the client will consider; however, early in the project the client may experience difficulty in identifying the solution constraints. For example, the question "Will you consider a package solution?" may receive a no answer because the client has little knowledge about packages. If the team briefly describes several packages that the team identified as possible solutions, the client can provide a better-informed reaction. The team has a due diligence responsibility—the responsibility to make certain that the client understands the available options.

Even at this early point, the team should consider the feasibility of the solution options. Some options may cost too much or take too long or require more skills than are available. As shown in the spiral model for system development (see Chapter 3), the team will revisit the issues of alternatives and evaluation at each stage of the project as additional information becomes available.

When the strict enforcement of a feature or constraint appears to produce a negative impact on the proposed system, the team may wish to revisit the requirements with the client. For example, the team may identify an attractive alternative that supports the strategic objectives that the client values, but the alternative lies outside the acceptable set of solutions; perhaps the cost is slightly higher than the client's budget limit or the solution is missing a desirable but noncritical function. Often clients will consider changing the conditions if the

team presents a good argument. Before raising questions, the team should collect the relevant information and do the analysis, that is, do the homework first.

Scope

The combination of the desired features with the constraints largely define the scope of the project. Figure 6.2 graphically represents one dimension of project **scope**—the acceptable systems that contain the desired features and fit within the constraints. Other aspects of scope include such things as the team's required skills, effort, and time to complete the project. Because many clients know what makes up a reasonable scope for a project, the project as defined by the client may encompass just the right scope for the team. However, the client may define a project with a scope that doesn't work. Scope problems include:

- *The project is too large and requires more effort than is available.* For example, a request by GB Video for a student team to redesign the entire information system including rental/return, purchasing, inventory, accounting, billing, accounts payable, personal payroll, advertising, and market research probably represents a scope beyond the effort available for most student teams in a one-semester time frame.
- *The project will take too long.* Some projects with a reasonable total effort requirement involve several sequential stages and take a long time regardless of how many people are assigned. For example, GB Video might ask a team to (1) conduct a mail survey of the GB customers to learn about video preferences and (2) create a procurement system that responds to the customer preferences determined from the survey. Either one or even both projects might fit within a reasonable scope except for the sequence constraint. The team must conduct and analyze the survey before the team can define requirements for the procurement system.
- *The project is technically too complex and/or demands skills the team does not possess.* Designing a high-volume travel reservation system or a loan scoring system may involve such very complex activities as high-volume transaction processing, business policies on booking, overbooking, cancellation, or decision making on establishing scores. Even a system that requires the use of a complicated programming language, network, or database environment may present technical and skill difficulties beyond the team's ability to master in the available time.
- *The project appears to address the wrong issues.* For example, GB Video might ask the team to set up a computer database to hold videotape and DVD rental records. GB Video proposes to continue creating written records during the rental process and then keying the data from the paper forms into a computer database. This scope probably offers few benefits. The more relevant scope might address capturing the data in electronic form during the rental activity and then storing it in the computer database.
- *The project is trivial.* For example, a project to design for GB Video a new paper file card to hold the information on each video gives a student team little opportunity to learn about project and system issues and to demonstrate mastery of the topics in the course.

All of the scope mismatches discussed in the preceding list will cause problems for the team. Too large or too complex projects will cause the team to either go without sleep and food for the duration of the project and/or fail to complete the project. A project that addresses the wrong issue may result in client disappointment at the end. The following equation represents a first approximation to the likelihood that a project will succeed:

$$\text{Project success index} = \frac{(\text{Time available} \times \text{People assigned})}{(\text{Size} \times \text{Complexity})}$$

For a given project, more time and or more people (i.e., more available effort) increase the probability of success. For a student project with a given time period and number of people, decreasing the project size and/or complexity increases the likelihood of success. Obviously a number of other factors, for example, people skills, management support, strategic relevance, and others, enter into the index.

Before finalizing a statement of work with the client, the team should carefully examine the scope issue with its manager. A wise team makes conservative commitments. Changes in scope can create problems with the client. The client may agree to scope reduction at the beginning of a project but may feel cheated, annoyed, or outraged if the team wants to reduce the scope near the end. The manager should review and agree to scope changes prior to discussion with the client.

Examples of Project Definition Materials

Figure 6.3 shows examples for the GB Video case of a project statement, strategic analysis, problem analysis, features, and constraints. Figure 6.3 provides only an illustration. Materials prepared by teams in practice may be more comprehensive. While the sections may remain the same for all field projects, the content and emphasis may vary. Each team should tailor their material to the client's specific problem.

WORKING WITH THE CLIENT

Starting with the project definition stage and at multiple other points throughout the project, team members will interact with the client. Because contact with the client represents a high-priority activity to begin work, the team should schedule an initial visit as soon as possible. Little other work can proceed until after one or more visits. If the team puts off the initial visit, the entire project may suffer.

The client wants to solve a problem with a goal of improving his or her organization. Clients are experts in the organization and its mission and operations, but frequently not experts in information technology. The client's interest in IT may extend only to its usefulness in solving the client's problem. The client has agreed to work with the team for one specific purpose—to help the team arrive at a solution to the problem. Working with the team may take a significant amount of the client's time and concentration.

FIGURE 6.3 Selected Project Definition Report Materials for GB Video**Project Statement**

GB Video wishes to automate their current manual system for processing videotape and DVD rentals and returns. Reporting will be handled by a separate system. The new system should address improved customer service and lower handling costs for each transaction.

Strategic Alignment

The team worked closely with Mr. Cosier to determine the strategic alignment for the project. Mr. Cosier has approved all statements on goals, objectives, and impacts of features.

Organization

Mr. Cosier, the owner of GB Video, stated, "The mission of GB Video is to serve the customer." The vision of GB Video is: In the markets where GB owns stores, become the leading seller and renter of videos and related supplies with the best selection of videos at competitive prices and with better customer service than competitors.

GB Video, with headquarters in Jackson, Oklahoma, operates three video stores in towns of about 10,000 people. Each store operates largely independently although the headquarters performs some functions for all three stores, for example, payroll, purchasing, and accounting. The company employs 37 people and realized revenues of \$1.5 million and profits of \$133,000 last year.

The stores operate from 10 a.m. to 10 p.m. except on Sunday when the hours are 12 noon to 10 p.m. Each store has a store manager and the store managers report to Mr. Cosier. Within a store, two assistant managers and a number of full and part-time clerks report to the store manager. The assistant managers are in charge of store operations on shifts when the manager is not on duty. The assistant managers and clerks perform any or all of the store functions as assigned by the manager on duty—serve customers, shelve new videos, shelve returned videos, send overdue reminders, etc.

Goals and Objectives

Discussions with Mr. Cosier and his staff people identified the following strategic goals and objectives. GB wants to increase profit to 10 percent or more of revenue and to increase sales by 5 percent per year after adjusting for inflation. The profit increase will result from reducing the costs of renting and selling merchandise and from economies of scale as the business grows. Mr. Cosier is in the process of renegotiating the leases for his stores and expects to save money for current and future stores with lower rent. The 5 percent sales increase will result from offering customers fair prices—at or slightly below those of competitors and providing faster checkout service and a wider range of selections than competitors. Mr. Cosier also wants to begin a targeted direct mail marketing effort using data about customers and rentals. The customer and rental data also will allow the purchasing department to do a better job of stocking the videos that are most in demand. He believes that these actions will

increase the number of members, the number of visits per member, and the average number of videos rented on each customer visit.

The GB Video goals and objectives are summarized in the table below. Some of the information in the table comes from the proposed system features discussion in the next section of this report.

GB Video Goals	Related Objectives
1. Increase GB profitability to 10% or more of revenue	1.1. Reduce the labor cost of renting and returning videos
	1.2. Benefit from economies of scale
	1.3 Lower rent for stores
	1.4 Reduce cost of nonreturned videos rented to unknown nonmembers
2. Increase sales by 5% a year or more after inflation adjustments	2.1. Competitive prices
	2.2 Faster checkout
	2.3 Tie video inventory to customer preferences
	2.4 Targeted mail advertising to members

Proposed System Contribution to Performance

Mr. Cosier states that the new system is mission critical for GB Video. Mr. Cosier plans to open additional stores if the new system results in improvements. The new information system should increase profits by reducing labor cost per transaction and eliminating the cost of videos rented to nonmembers and not returned. Faster checkout service and better selections can lead to increased customer satisfaction, more members, and higher sales. GB Video looks at labor costs, profits, and revenues per store as major performance measures and at the total number of active members and the number of rentals per member. The accompanying table summarizes the impact of proposed system features on objectives. The impact target data in the table are derived from discussions between Mr. Cosier and the team and approved by him.

The new system will not affect the rental cost of stores. The system is not involved in setting rental prices. As a result, objectives 1.3 on store rental costs and 2.1 on pricing do not appear in the table and are not discussed in subsequent analysis.

Objective	Measure	Feature	Impact Target
1.1. Reduce the labor cost of renting and returning videos	Labor cost	Automate the rental/return processes	\$50,000 per year in labor cost savings
1.2. Benefit from economies of scale	Increase in revenues versus increased costs	Automated system handles increased volume with little or no new cost	Increases in costs equal to 85% or less of increased revenues
1.4 Reduce the cost of nonreturned videos	Lost video costs	System must confirm member before renting	\$5,000 per year
2.2 Faster checkout	Checkout time	Automate process	Decrease from an average of 4.5 to 1.5 minutes
2.3 Tie video inventory to customer preferences	Increase in sales; increase in members	Rentals database available to purchasing department	5% increase a year
2.4 Targeted mail advertising to members	Increase in sales	Rentals and customer databases available for use	5% increase a year

Project Success Criteria

Discussions with Mr. Cosier led to the following measures of success for the project. Mr. Cosier will consider the project a success if the team:

1. Completes the project on time. Since the team will not handle implementation, success is defined as submitting a complete final report by April 23.
2. Identifies and fully specifies a solution that provides the features desired by the client within the client's constraints, or demonstrates clearly that no such solution exists. If no solution can be found, Mr. Cosier wishes to know this information as soon as possible so that he can consider modifying the requirements.

While Mr. Cosier feels strongly about the profit and sales goals, he understands that many factors other than the project influence profits and sales. He will consider the project a success if the team accomplishes items 1 and 2 above.

Features for the Proposed System

The team identified the following features based on discussions with the client representatives. As shown above, these features contribute to the organizational goals identified in the strategic analysis—increased profits and sales.

The client wants the proposed rental/return system to include the following functions:

1. Membership
 - a. Collect data and store data on new members.
 - b. Update data for existing members.
 - c. Issue a member card to members.
2. Rental
 - a. Rent tapes only to members.
 - b. Create and store a rental record with identification of member and video.
 - c. Adjust an inventory record to reflect the rental.
 - d. Issue a receipt to the customer/member.
3. Return
 - a. Update the rental record and the inventory record to reflect the return.
 - b. Calculate the overdue charge if any.

The client wants the processes to accomplish the above functions changed so that they require less clerk time. The client also requests that the proposed system contain the following features:

1. Automate the entry of video ID and member number. The client asked the team to look at bar code scanning.
2. Automate the entry of the current date/time for rental and return.
3. Reduce the time the clerks spend working with the paper files—the files of return forms, videos, and members.
4. Eliminate duplicate manual data entry and storage, for example, entering the rental date and return date on both the video card and the rental form.
5. Eliminate the cost to manually send data to accounting.
6. Reduce the cost of separate credit card and rental transaction processing.
7. Move all reporting functions out of the rental/return system.
8. Automate the sending of overdue notices to customers as part of the rental system.
9. Provide rental data for a data warehouse to be used by purchasing.
10. Provide rental and customer data for a database for marketing department.

Constraints

Mr. Cosier states that GB Video will consider solutions that will realize a payback in two years or less and cost less than \$200,000 up front to acquire and implement. The system should be in full operation in one year if possible.

GB is willing to consider changes in function and organization if the changes provide significant benefits to GB. In particular, GB wishes to transfer reporting and overdue notice functions to the accounting group.

Adequate air-conditioned space in the headquarters exists for a server and network room. GB will acquire the equipment if that is the best alternative. Since GB currently owns no computers, interoperability with current hardware and software is not a problem.

GB does not wish to continue with the current system unless all computer-based alternatives are prohibitively expensive, i.e., outside the \$200,000 limit. The procurement options that GB will consider include:

- 1. Package System.** Search for a package system that will handle the rental and return tasks to the client's satisfaction. GB prefers to purchase a package as long as the cost to purchase and implement the package including infrastructure does not exceed \$200,000.
- 2. Contract for Service.** Contract with an application service provider (ASP) to provide and operate the rental and return system. Mr. Cosier will consider contracting with an ASP if a suitable package is not available or cost-effective.
- 3. Contract for a Custom Package.** Contract with an IT consulting company to create a custom package program for the rental and return system. Mr. Cosier will consider this alternative if a suitable package is not available or cost-effective.
- 4. Build the System In-House.** Mr. Cosier states that GB Video has no in-house capability to build a system. Only if all other alternatives look unsatisfactory is he willing to consider developing such a capability. He believes that the cost probably is prohibitive.

Page 5

In return, the team accepts the responsibility to (1) use the client's time efficiently and to (2) deliver as much value to the client as possible—in the form of such deliverables as project definition, proposed system specifications, a recommended solution, and a proof of concept demonstration model. The following guidelines apply to all of the contacts with the clients for any reason and should help the team to establish and maintain a good relationship with the client. Note, however, that no guideline works with all the people all the time. Always watch the client and observe the client's reactions to what is happening. If the client expresses or shows any signs of boredom, disagreement, annoyance, anger, or distress, try to identify the problem and use a different approach.

Professional Behavior

Each team member should demonstrate to the client that he or she is an "information systems professional." Team members should behave in the same manner as a competent and successful member of a major consulting firm. **Professional behavior** includes such conduct as the following:

- *Demonstrate a professional regard for the role of the client.* Try to react only to the content provided by and the role of the client. Try to avoid letting the client's personality, appearance, and manner influence your views.

In project work, the client is, within reason, always right. Do whatever you can to provide good information and analysis to help the client explore issues and alternatives. But when the discussion is over, accept and follow the client's decision even when you think it is incorrect. If the client insists on an unreasonable request or gives too little time or attention to your project for it to succeed and you have tried every polite and reasonable way to bring about a change, contact your manager. If the client asks or implies that the team take any action that appears even marginally unethical or illegal, contact the team manager immediately.

- *Submit professional deliverables.* Professional deliverables contain correct and complete content, and consist of neatly printed materials with clean, clear diagrams and attachments, good writing, and clear, consistent format. Always use spell check and grammar check, but review carefully what grammar check wants you to do. Note also that spell check is not a *meaning* check: It misses all instances of using a correctly spelled word with an incorrect meaning, for example, "work" in place of "word," "he" in place of "the," or "the" in place of "they."
- *Look like a professional.* Appropriate professional appearance covers a broad range in the modern "business casual" world. A good guideline is to dress as well as the best-dressed client in the meeting. If you plan to visit with a foreman who is in overalls and the president who is in a suit, dress to match the president. When in doubt, use conservative business attire: suit, coat and tie, a business dress or pants suit. Unless the client states otherwise, a consultant can wear business attire to visit any client representative, even a foreman in a dirty, hot shop; but both you and the foreman may feel more comfortable when you wear casual clothes.
- *Protect your **professional integrity**.* Regardless of what the client or your team members say or want, do not participate in any activity that you know or suspect is a violation of ethical behavior, is illegal, involves a significant risk of injury or death, or might expose you to serious personal liability. As soon as you suspect such a situation, do not participate and contact your manager immediately.

Prepare for a Visit

Once a project is assigned, the team can contact the client and arrange to meet. The first visit generally will follow an interview format but also may involve document collection and observation. Initial impressions are important. When possible, the team uses the first visit to create a serious, professional, work-focused model for future visits. The first visit should encompass as much work as possible. Just showing up and saying hello wastes both the client's and the team's time. During the first visit, the team should outline and, if possible, schedule the other planned visits including the project definition and final presentations. The team should explain any schedule constraints to the client and ask for his or her help in meeting the schedule.

Successful client visits start with good preparation. Normally, the team will meet to prepare before a visit, especially the first several visits. A team that shows

up at a client meeting without any preparation looks unprofessional, wastes the client's time, and wastes the team's time. As part of the preparation, team members should think carefully about what information to collect on each visit. The client may have little understanding of the information that the team wants. The team members can make the best use of both their own and the client's time by developing a clear map of the information they wish to collect. Looking at the information needed to (1) answer the questions in the introduction to this chapter and to (2) prepare the final report and final presentation provides a good starting point on what information to collect.

For most projects, the team will begin by collecting information to prepare materials on the following:

1. Project plan.
2. Problem statement.
3. Strategic alignment.
4. Goals and features for the proposed system.
5. Procurement options that are acceptable to the client.
6. Constraints.
7. Current situation.
8. Statement of work.

The team should review each of these topics before the first client visit and prepare a list of questions and requests for information. Much of the aforementioned information is needed for the project definition report and presentation early in the project. Other information depends on the procurement options selected by the team. For example, a team that decides to purchase a solution may collect extensive information on packages while a team that decides to build may collect detailed data and process information. The basic message is simple: Identify what the team needs to do and then collect the information to support the actions. Collecting all the information that the team happens to encounter and then looking at what needs to be done can waste a lot of time and jeopardize the project.

Other guidelines for advance preparation by the team include these tasks:

- *Learn as much as possible about the organization, problem, and client.* The team should show respect for the client's time by learning from available sources before asking the client questions. For example, the organization may operate a Web site, and articles about the organization or about the industry may be available in trade publications. The team should study the readily available knowledge about the organization and the current situation before the first interview with the client.
- *Telephone or e-mail the client to schedule a visit.* Unless the client tells the team otherwise, avoid showing up without an appointment or arrangement. To do so may suggest that you think the client's time is less valuable than yours. Schedule visits as far in advance as possible. In the first meeting with the client, the team can outline the expected pattern of visits for the course of the

project and actually schedule some if not most of them. However, only schedule a visit if a clear and necessary reason for the visit exists. Offer the client the option to cancel a scheduled visit if it no longer appears necessary.

- *Work on building a good relationship with the client's secretaries and/or assistants.* An assistant may manage much of the team's relationship with the client. Try to learn names, telephone numbers, e-mail addresses, and other important pieces of information. Call staff members by name when you see or talk with them. Thank them for their help. Staff members can assist you greatly or make life difficult if they so choose.
- *Always prepare an explicit, preferably written, plan and agenda for each client visit.* The team can prepare the written plan in rough draft form or even on the back of an envelope. List the purpose of the meeting, questions for which the team wishes to obtain answers, materials that the team wishes to request, and the agenda, which states who will do what and when at the meeting. If possible, send the agenda to the client prior to the meeting.
- *Each team member who is present at a client meeting should have at least one role: meeting organizer, agenda presenter, question asker, listener and note taker, materials collector, or other.* Many team members will perform in several roles. If a team member has no role, the client will wonder why he or she is there.
- *The one person designated in your team contract or update thereof as the client contact person should take responsibility for coordinating the visit arrangements with the client and communicating the arrangement to the other team members.* If several people contact the client to arrange a visit, a high likelihood exists of conflicting or confusing messages. The visit coordinator should make sure that all team members, and the manager if present, receive clear instructions on how to get to the client's office and who or what to ask for (person, room, etc.).
- *If possible, explain clearly and briefly to the client in advance why you are coming and what you expect to accomplish.* Advance notice gives the client the opportunity to collect materials and information, thereby saving time and avoiding possible embarrassment. Providing the client with a copy of the agenda, questions, and so forth, also can help to focus the meeting.
- *Use e-mail and/or the telephone in place of more time-consuming and disruptive in-person visits when appropriate.* Avoid visits to answer a simple question, for example, "How many workstations are available at the Help Desk? "or" Is Tuesday April 21st at 2:00 p.m. an acceptable time for our final presentation?"

Make a Visit

Visits give the team the opportunity to observe the organization and build a good relationship with the client. Visits provide a greater level of flexibility than e-mail or telephone contacts. If the client mentions a form, report, or document, the team can ask to see it and/or obtain a copy. Clients also tend to provide more information during in-person visits than they do using e-mail or the telephone. The team should take every possible precaution to see that each visit gains the needed information for the team and leaves the client feeling good about the relationship.

Some guidelines for visits include:

- *Arrive on time.* Allow for getting lost or stuck in traffic; it happens more than you may think. If you arrive more than 15 minutes early, wait outside until the 15-minute mark; use the time to review your preparation for the visit. If you must arrive late due to unforeseen circumstances beyond your control, try to notify the client and/or your team.
- *Assure the client at the beginning of each meeting that you will safeguard any information you acquire.* You will use the information only for the project purposes and will not reveal it to any person not affiliated with your team. Use extra care to honor this pledge unless the client authorizes other use of the information. Many companies consider the content of some of their information systems as proprietary. Do not argue or complain if the client declines to give you certain pieces of information. Ask for alternatives, for example, a sample report in place of a real one.
- *Ask permission in advance if you wish to record the client's remarks.* Even if the client agrees, watch for signs of any problem especially in an informal meeting with a single client. Sometimes recording makes the client nervous and reluctant to talk openly about the situation. In a more formal meeting with a number of clients present, clients already tend to guard what they say, and recording is less likely to inhibit discussion.
- *Treat the client with professional respect and respect the client's time.* Ways of showing respect include:
 - Have a well-thought-out plan and follow it.
 - If the client demonstrates a lack of understanding of technology or some other issue, do not make any kind of a demeaning remark or expression—grimace, roll your eyes, sigh, or other.
 - Always allow the client to finish a sentence, point, or thought. If the client introduces unrelated, irrelevant, or personal topics, listen politely until the client stops talking or at least stops to take a breath. Then try to return to your agenda.
 - If the client makes a statement you believe is incorrect, ask for clarification politely and in a neutral way. For example, “Could you explain; I am not sure that I understand.” If the client still appears wrong, do not argue with, contradict, or challenge the client. Check other sources.
 - Keep your discussion focused on topics related to the project. The team members should not introduce unrelated or personal topics. The client may not want to spend his or her time learning how you spent your summer vacation or hearing a joke you like.
 - Do not ask irrelevant or personal questions or questions with obvious answers. Many inappropriate questions result from nervousness and confusion caused by lack of a plan for the visit. Examples of inappropriate questions might include: “Do you like your boss?” (personal and possibly embarrassing); “Who was the architect for this building?” (irrelevant); or when introduced to the CIO asking, “Do you work in IT?” (obvious).

- *Encourage the client to interact.* Some clients tell more than the team wants to know while others volunteer nothing beyond a direct answer to questions. If the client does not interact on his or her own, ask questions or ask the client to comment on the team's remarks or materials. The team then can ask follow-up questions on the client's remarks.
- *Demonstrate to the client that you are listening.* Show interest by sitting up and looking up often at the client. Assign some team members to take notes and others to focus on the conversation.
- *Take good and complete notes or record the conversation if the client agrees.* Showing up to ask the client about things he or she already told you a week ago shows disrespect for the client's time and looks unprofessional.
- *Use feedback to demonstrate that you absorb what the client is saying.* You can provide feedback by:
 - Asking the client to elaborate on what was just said, for example, "Can you tell us more about what the order clerk does when the customer has no record."
 - Repeating the statement: "Do I understand correctly that the order clerk cannot process the order if the customer has no record on file?"
 - Making a change to reflect the client's comments on your flip chart or on the whiteboard if they exist.
- *Manage the end of the visit.* Establish how much time the client wishes to spend when the team sets up the visit. Confirm with the client at the beginning of the meeting how long he or she wishes to spend with you. When the time is up, volunteer to leave even if the client does not bring it up. If client wants you to stay longer, the client can tell you. If the client declines to offer you more time or when you have the information that you want, thank the client and leave promptly. Once the visit reaches the end point, your departure should take only a minute. Do not stand at the door for ten minutes with a last few questions. To paraphrase Yogi Berra, "When it's over, it's over." You can ask to schedule another visit if more time is needed.

To summarize these guidelines, behave as a professional with a courteous and respectful manner, plan your visits, demonstrate you are listening, and stay focused on business during visits.

INFORMATION COLLECTION APPROACHES

During this stage the team must collect at a minimum the information needed to prepare the project definition deliverables defined in the plan. These products clearly include the ones defined in this chapter and in Chapter 7. In addition, the team can save time and work by collecting as much as possible all of the information needed to produce a final report and presentation for the client. In other words, team members need to look ahead to assure that they collect the right information. The team's plan for information collection looks carefully and in detail at the deliverables.

Collecting the information that the team will want probably will require multiple visits and approaches. The team can utilize various approaches to collect information including: (1) interviews, (2) document collection, (3) observation, and (4) surveys. Often these approaches are used in combination. The team may wish to use one or more of these approaches to collect information from both the client and others, for example, vendors and/or from similar organizations and applications. These other organizations may have found a solution to the problem that the client posed or may know what doesn't work.

Interviews

Interviews with clients, users, senior managers, systems people, and others offer a widely used approach to collecting information. Completely unstructured interviews, ones with no prior plan or agenda, waste both the client's and the team's time. Before any interview, the team should prepare an explicit agenda and a list of questions and requests. The team may wish to give the client a copy of the agenda in advance. Guidelines for the interview include those below.

1. *Start with a general open-ended question for each area of interest, for example, "Please describe how the current system works when a customer wants to rent a tape," or "What are your goals for the new system?"*
2. *Follow up with more specific questions to clarify and expand on points the interviewee made or to cover areas that the interviewee did not address, for example, "Can a customer without a credit card become a member?"*
3. *If the interviewee mentions a form, report, or other document, ask for a copy.*
4. *Avoid leading questions that imply a right answer. Examples include, "Did you follow good practice and prepare a project plan?" or "Of course, you have an IT strategic plan, don't you Mary?" The client may take offense or may answer incorrectly in an attempt to save face. A better question is, "We want to make sure that we understand your practices. Do you have plans or guidelines that relate to IT?"*
5. *Be opportunistic. If the interviewee mentions a topic or issue that is not on your question list but seems relevant, ask questions to explore it further. A plan gets you started, but should not constrain you from developing new areas.*
6. *If the interviewee does not know the answer to a relevant question, ask for suggestions on where or how you might get the answer.*
7. *At some point, ask clients about procurement options that are acceptable to them. For example, "Which build and purchase options for a proposed system can the team consider?" If asked about your thoughts on good solutions by a client, say that you wish to complete your analysis before making any recommendations. The purpose of an interview is to collect information, not to give advice to the client.*

Group Interviews

Group interviews may facilitate the information collection process. Often one person may know only part of how things work. He or she may be unable to answer some questions or may answer incorrectly based on casual knowledge or perceptions outside direct experience. Interviewing a group of people may

mitigate some of the problems. For example, an interview with a group consisting of users, clients, and system analysts may provide a more complete picture than interviews with each separately. If different people give different answers to the same question, a group interview may help to resolve the differences. The interviewer must take care to remain neutral and not to appear to side with one faction when conflict arises. The client may decide to arrange group interviews or the team may suggest them.

Focus groups offer another form of group interview. To use a focus group, the team arranges with the client to interview a group of people with similar responsibilities and authority, for example, a group of order clerks, without any supervisors present. Explain to the group members that you plan to share the comments with the client but will not identify who said what. With the presence and reinforcement of co-workers and in the absence of superiors, people often will talk more openly about problems and current practices. When reporting the comments from the focus group to the client, remember to honor your commitment and keep all comments anonymous.

All group interviews present possible group interaction hazards. One or several loud, strong, and assertive people may get the group to espouse a view that many of the other people in the group do not share. When the team suspects that this problem is occurring, try to arrange to talk with some of the less assertive people individually.

Documents

In most current situations, the team can find **documents** that describe or illustrate how things work or at least how they should work. Examples of documents include the IT and/or the organization's strategic plan, organization charts, procedure manuals, policy statements, memos, data input forms or screens, reports, instructions for system users, file formats, data schema, flow charts, other diagrams, system documentation, and reports of auditors and internal or external consultants. Ask the client if any of these documents exist and if you may have copies of them. The client may consider some documents as proprietary and decline to release them. Attached documents can add greatly to your narrative description of the current situation.

Observation

Even the best descriptions of how things work leave unanswered questions. In addition, people often follow different processes than the ones described in documentation or by their managers. For example, telephone order takers probably know ways to use the system that the manager or analyst never considered and ways to compensate for errors that the manager or analyst does not know about. If possible ask to observe and talk with system users at work. The team may learn a lot by observing the operations of an organization that runs a system similar to the proposed system. If permission is given, take special care not to interfere with ongoing operations.

Observation by an outside group, even a group of students, may make system users nervous and may change their behavior. People who seldom or ever

follow the written procedures may do so when under observation by outsiders. The team should take extra efforts to avoid saying or doing anything that might imply criticism or evaluation of the users. Take pictures or recordings only with permission.

Surveys and Sampling

If a user or client group contains a number of people, interviewing every member may be impractical and unproductive. For example, an airline or car rental company may employ thousands of reservation takers. On the other hand, the views of one or several people may differ substantially from those of the full group. One alternative is to interview a representative **sample** of group members, but even a reasonable sample may require a large amount of interview time. In this case, a survey may offer the best choice.

Normally project team **surveys** are used informally to discover general views or trends. In a few cases, the team may wish to apply statistical analysis to determine confidence levels on the results. With a well-designed survey, the team may discover different views within different subsets of the group. For example, users in small offices may encounter different issues than users in large offices.

The team starts by preparing a set of questions or the survey form. Short surveys, one page or so of questions, tend to work better than long surveys. With long surveys, many people lose patience and either decline to complete the survey or rush through giving random answers. Test the survey on three or more typical survey participants before starting the main survey. Questions that seem obvious to the team may be totally mystifying or mean something quite different to the participants.

If feasible, telephone surveys probably provide the clearest picture. With a telephone survey, the team gets to pick and/or control the people who actually respond. If a question confuses the participant, the team member can try to explain and clarify. Participants also are more likely to answer thoughtfully with a live human on the other end. However, telephone surveys can use up a lot of team member time.

For an organization, a written survey form probably is the easiest format. With the active support of and a cover letter from management, the team may get reasonable return rates, perhaps 15 to as much as 80 percent. Written surveys often tend to reflect bias—the people with strong feelings respond at a higher rate than the typical people. E-mail surveys increasingly get lost among the flood of spam unless the organization has effective spam controls and/or the survey comes from a manager. For customers or other groups outside the company, the team can try a telephone, Web-based, or e-mail survey. The participation rate for outsiders with e-mail and Web-based surveys, however, will tend to be low.

Summary

Organizations sometimes say they want better information systems. Actually organizations want better performance with respect to such overall organizational values and goals as higher stock prices, more profit, higher sales, lower costs, and more satisfied customers. To succeed, an IT team needs to understand

and think like the client. In well-run organizations, managers at all levels and in all areas of the organization are constantly reminded to focus on strategic goals or on critical organizational values. Strategic alignment of IT activities means focusing IT activities to support organizational values.

A project team should understand and support organizational values for the following interrelated reasons:

- To gain organizational acceptance and credibility.
- To obtain and retain financial sponsorship.
- To justify the recommendation.
- To make good ongoing project decisions.

The team collects information and prepares an organizational case to establish and maintain the alignment of the project with organizational values. The organization case for strategic alignment rests on three components: the organization, goals and objectives, and project contribution. Some of the organization attributes that may hold importance for the project include vision, mission, culture, organization charts, authority and responsibility, available resources, legal and regulatory issues, security, glossary of key terms, and a description of current operations.

A goal is an expression of a commitment to a state that the organization wants to achieve, for example, to become more profitable is a goal for every airline. Typical high-level goals refer to such areas as profits, cost-control, revenue enhancement, sales, quality, service, and customer satisfaction. An objective is a concrete, measurable action that supports that goal, for example, increase revenues by increasing aircraft utilization with faster turnaround and better scheduling. The real issue of alignment is to assure that project outcomes support the goals and objectives of the organization.

The final and critical step to strategic alignment consists of identifying the contribution of the project to the strategic objectives of the organization. The team either finds or formulates objectives that (1) lead toward the selected goals and (2) are facilitated by the project. Often, the objectives target improving a part of the organization that does not function as well as desired. In this event, the team identifies the problems and adds or constructs features for the proposed system that will improve performance with respect to objectives. This process is known as “performance-oriented design.”

The team should identify specific, tangible performance measures that the proposed system will impact. The performance measures must relate directly to the objectives. Finally, the team identifies measures of success for the project. The success measure may involve the performance measure; the project is a success if sales costs are reduced by 10 percent. Often measures of success may involve several parameters, such as to reduce costs by 10 percent and to complete the project on time and on budget.

The project definition report contains the team’s understanding of the client’s problems and requirements in an organizational context. The project definition report is “client-centric” in that it reflects only the client’s views and desires. Many, if not most, projects take longer and/or cost more and/or produce less

than expected because the team does not identify fully and correctly the client's desires and requirements. The difficulties may result from the client's inability to express requirements or from the team's lack of focus and effort to understand the client or from a combination of both. The report normally contains the following materials:

- Title Page
- Table of Contents
- Executive Summary
- Introduction
- Project Definition
 - Introduction
 - Project Statement
 - Strategic Alignment
 - Functions and Features
 - Constraints
 - Current Situation (see Chapter 7)
- Proposed System (stub)
- System Delivery (stub)
- Appendix A. Statement of Work (see Chapter 3)
- Appendix B. Documents supplied by the client
- Other Appendixes as needed

The team acquires and analyzes information in the project definition stage to answer the following questions:

1. Who is the client? Who is the person or group with the authority to sponsor or define the project?
2. What does the client want? What are the scope, goals, features, and structure for the proposed information system?
3. Can the proposed system solve the client's problem? Can it eliminate or reduce the problems and/or achieve the client's goals?
4. What does the organization value? What are the core competencies for the organization and what are the key measures of organizational performance?
5. How will the proposed system add value to the organization? How will it enhance or contribute to the performance measures of the organization?
6. What are the constraints? What resource, procurement option, and other limits does the client wish to set for the team?
7. What can the team learn from the current situation? What aspects of the current operations relate to or can contribute to the design of the proposed system?

To gain answers to these questions, the team interacts with the client. The team must maintain a solid professional working relationship with the client by following such guidelines as to submit professional deliverables and to behave with professional integrity.

The team should prepare for each visit by developing an explicit statement of purpose for the visit, a list of questions, and an agenda. A good plan makes efficient use of both the client's and the team's time. Other guidelines for visits include to schedule in advance, arrive on time, assure the client that the team will safeguard information, ask for permission to record the meeting if so desired, encourage the client to interact, demonstrate to the client that the team members are listening, treat the client with respect, and manage the end of the visit.

The team can utilize four approaches to collecting information: (1) interviews, (2) document collection, (3) observation, and (4) surveys. Often these approaches are used in combination. Interviews offer a good method to obtain information, but effective interviews require structure and planning. The team can interview the clients individually or in groups. The team collects any relevant documents that are available. The team can learn about or verify how the current process works by directly observing the current operation. The team can use surveys to obtain information from a large group of people.

The team structures the information collection to obtain the information required for project definition: project statement, strategic alignment, functions and features for the new system, and constraints. The functions, features, and constraints narrow the set of all possible solutions to the set that meets the client requirements.

Key Terms

constraints, 207	objectives, 197	project statement, 203
documents, 222	observation, 222	project success, 198
features, 198	organization, 204	recommended solution, 196
focus groups, 222	performance, 206	sample, 223
functions, 206	performance measures, 198	scope, 209
goals, 197	performance-oriented	strategic alignment, 197
group interviews, 221	design, 198	structure, 206
impact of a feature, 196	procurement options, 208	surveys, 223
information collection, 195	professional behavior, 215	vision, 198
interviews, 221	professional integrity, 216	
mission statement, 198	project definition, 194	

Review Questions

- For a strategic analysis of an organization, answer the following:
 - What are the basic purposes of the analysis?
 - What are the strengths and weaknesses of published mission statements?
 - When should you focus on strategic alignment with the sponsor instead of the entire organization?
- Define the following terms:
 - Vision
 - Mission
 - Goal
 - Objective

3. Using the GB Video example, either quote or infer the following:
 - a. What is the problem GB Video wants to solve?
 - b. What is GB Video's vision of itself?
 - c. What is GB Video's mission?
 - d. What GB Video goal(s) are relevant to the problem?
 - e. What GB Video objective(s) are relevant to the problem?
 - f. For each objective, what measures are important?
4. For features and constraints as discussed in the text, answer the following:
 - a. What is the difference between a feature and a constraint?
 - b. Why do teams often not try to produce an optimal solution?
 - c. What is the difference between a mandatory and a desirable feature and/or constraint?
 - d. When your client states, "I want better performance," what may the client actually want?
5. What are four approaches to collecting information?
 - a. When is it appropriate to use each?
 - b. What are the strengths and weaknesses of each one?
6. You are preparing for an interview with your client.
 - a. What sources of information can the team use to prepare for the visit?
 - b. How can a team demonstrate that the members are a group of information systems professionals?
 - c. What are the roles that the team should assign to members and why is it important to have roles in an interview?
 - d. What are guidelines for setting up meetings that respect the client's time?
7. For the team to manage client relationships:
 - a. Name some events, incidents, or activities that will damage the relationship with the client.
 - b. What are some important guidelines to remember when interacting with the client?
8. When arranging and making visits with clients,
 - a. List guidelines for successful visits with the client.
 - b. What preparation should the team make for each visit?
 - c. What are some behaviors team members should avoid?
 - d. Contrast the role of a client and a team manager.
9. For a project definition report,
 - a. What are the components?
 - b. For each section of the report, write a one-sentence description of what that section should do.
 - c. For each section of the report, describe the target reader.
10. To determine the right project scope,
 - a. When bounding a given project, what are some indications of possible scope problems?
 - b. What should a team do when they expect that a project scope is too large or too small?

Critical Thinking Exercises

Individual Exercises

1. Based on the GB Video illustrations in Chapter 6,
 - a. Prepare the executive summary for the GB Video project definition report materials presented in Figure 6.5 in this chapter.
 - b. Where will the team go to gather information on the GB Video project?
 - c. What method of information collection will the team use for GB Video?
2. The team has been assigned to work up a series of banking summary reports for year-end reporting requirements to the FDIC. Upon visiting with the client, the team ascertains that the reports come from an IBM AS-400 and are written in RPG, a mainframe COBOL environment, and from an Oracle Database. None of the team members have RPG or COBOL skills. How should the team approach a scope discussion with your manager?

Group Exercises

1. The team has been asked to create an automated appointment system for your college's advising system. Use team knowledge to answer the following questions.
 - a. What is the business problem?
 - b. How will a solution to the problem contribute to the client's mission?
 - c. What are specific objectives that the system will address?
 - d. What measures of performance do these objectives suggest?
 - e. How will the proposed computer system help solve the problem?
2. The team has been asked to interview the dean of the business college concerning an alumni database for all graduates of the college.
 - a. Write up questions to ask the dean.
 - b. Who else should be interviewed for this project?
3. The team has been assigned to the staff of *Monday Night Football*. Management wants the team to set up a quick retrieval warehouse for team and individual statistics as well as personal anecdotes on the players. The announcers need this information during the broadcast of the game and must be available with a series of triggers for certain types of information.
 - a. What information collection system will the team use? Justify your choice.
 - b. Who will the team interview to get the specifications? Use imagination on this issue.
 - c. Who is the client?
 - d. Who are the users?
 - e. Prepare the project statement and the strategic analysis part of the project definition report for the above project.
4. Using the project defined in Chapter 3, Group Exercises, problem 4, prepare the following:
 - a. Project statement.
 - b. Strategic analysis of the organization.
 - c. Goals and features for the proposed system.
 - d. Constraints of the new system.

Reference

Kendall, Kenneth E.; and Julie E. Kendall. *Systems Analysis and Design*, 5th ed. Upper Saddle River, NJ: Prentice Hall, 2002.

Chapter Seven

Learning from the Current Situation

Chapter outline

Introduction

Information Collection

Current Situation Narrative Model

Description of Current Operations

Physical and Organizational Infrastructure

Problem Analysis

Retention and Change Analysis

Correctness and Completeness with

Multiple Representations

Current Operation Graphical Process Model

Guidelines

Process Model Metadata

Current Operation Graphical Data Model

The Project Definition Presentation

Completing the Project Definition Stage

Summary

Key Terms

Review Questions

Critical Thinking Exercises

Individual Exercises

Group Exercises

INTRODUCTION

The team normally analyzes the current situation to learn more about the environment and requirements for the proposed system. The **current situation** contains the organizational functions or activities that the organization currently performs in the areas within the project scope. Sometimes a current information system, either a manual or a computer-based one, will exist to support some of the organizational functions, but the client wishes to modify or replace the existing information system. In other cases, the client wishes to devise an information system to support an organizational function or a set of functions that are not supported at present, a situation in which a current information system with processes and data to support the organizational function does not exist. Even when no current system exists, the team can conduct an analysis to determine how the proposed system will integrate with the current situation.

The current situation analysis examines (1) the current operations—the existing events, inputs, processes, data, and outputs associated with the organizational functions within the scope of the project; (2) the existing organizational and physical infrastructure to support the operations; and (3) problems with the current operations and infrastructure. For many projects, the information system content model with data, process, and infrastructure provides a useful framework for the analysis of the current situation. The team may model data, process, and infrastructure in two representations: narrative and graphical. The multiple representations help the team and the client to check carefully for completeness, correctness, and consistency. Graphical models offer a convenient way to organize and cross check data and process information. Graphical models also provide a standard format that may facilitate communication between team members and sometimes with clients; however, many clients will prefer a narrative model.

The goal of current situation analysis is to obtain information to answer the following questions:

- What happens in the current operation? What and how do events, processes, and data interact as the current operations start, proceed, and end?
- What existing physical and organization infrastructures support the current operations?
- What problems result from the current operations and infrastructure? What things happen or do not happen that the client considers undesirable?
- How do the problems in the current situation affect the performance measures identified during the strategic alignment analysis?
- What aspects of the current situation may or must appear in the proposed system solution? What operations, problems, and infrastructure carry over to the proposed system?
- What aspects of the current situation should change in the proposed system solution? How should operations, problems, and infrastructure change?

The level of detail and effort devoted to the analysis of the current situation should vary from project to project. When the new system represents a modification of a current system to introduce new technologies, add new features, or correct some existing problems, the team may conduct a more detailed and in-depth analysis of the current situation as part of the proposed system activities discussed in Chapter 8. When the new system bears little resemblance to the current one, the limited analysis of the current situation prepared for the project definition will suffice. In a few cases, the current operations may offer little or no insight on the functions that the client wants for the proposed system.

At the end of the analysis of the current situation, the team should have information to (1) identify the infrastructure, data, and process that will remain and that will change with the proposed system; (2) demonstrate to the client that the team understands the current situation; and (3) refine the features and constraints for the proposed system. The goal is to obtain enough information to provide “satisfactory” answers for defining the proposed system, not to learn everything possible about the current situation.

During the analysis of the current situation, a team may prepare a narrative model plus graphical data and process models. The narrative model describes the current situation in natural language, but follows a format to encourage completeness and facilitate communication between team members. The data and process models present a conceptual-level graphical view of the information-related activities including the actual physical data stores, data flows, and processes that exist in the current operation. The current operation models may encompass more or fewer data and functions than the new system will include.

The analyst must decide in what order to conduct the analysis and prepare the models. Many analysts start the modeling process for the current situation with what they see as the most “natural form”—the narrative model. Clients generally describe the current situation in narrative form. In many situations, the operations portion of the narrative model translates easily into DFDs. Some analysts argue that the graphical models, data flow diagrams (DFDs), entity relationship diagrams (ERDs), or similar, are the appropriate starting point. Some teams assign the tasks to different members and prepare the various models simultaneously. This approach may lead to the team spending extra time to coordinate the models.

Whatever the starting point, the goal remains the same—the narrative and graphical models together should give a complete, correct, and consistent view of the current situation. Achieving this goal probably will require several iterations, for example, changing a DFD to correspond with the narrative or changing the narrative to reflect some newly discovered processes and flows on the DFD.

INFORMATION COLLECTION

During the strategic analysis and the determination of features and constraints for the proposed system, the team relies mainly on interviews or discussions with the client supplemented perhaps by such documents as a strategic plan, if available. To learn from the current situation, the team can continue to talk with the client but also may wish to apply some of the other information collection approaches described in Chapter 6. For example, the problems with the current situation described by the client may not mean much until the team directly experiences or at least observes them. And the best way to gain the knowledge to prepare the narrative and graphical models is to observe or participate in the current situation. Sometimes the client may express reluctance for the team to directly observe or participate. Often the client welcomes direct involvement from the team or from one or two team members.

When relevant, the team members should ask to:

- *Observe the current situation.* Most organizations will allow the team to observe unless the situation is hazardous or there are security issues. In the GB Video example, observing means going to a store and watching clerks process rentals and returns for customers.
- *Obtain forms, reports, and other related documents.* Some organizations have formal documentation for existing systems. The team always should ask for systems documentation, but such documentation seems unlikely to exist at

organizations like GB Video. At GB Video and most organizations, the team can obtain copies of all the forms used in the system including the reports that are prepared, overdue notices sent out, and other forms. GB Video and many organizations may have instruction or procedure manuals for system participants. Most organizations will let the team examine the materials that are available subject to proprietary, privacy, or confidential information limits.

- *Talk with users.* Users often express views and insights about the current situation that the client neglects to mention or may not know. Most of the time, the client will allow the team to talk with employees. Although the team should inquire, the client may or may not want the team to talk with such external people as customers or vendors. The client might allow the team to survey customers and vendors, another way of talking with them. At GB Video, users include clerks, vendors, and customers.
- *Participate in the current situation.* Actually using the system often provides the best possible insight on the problems with the current situation. The team certainly should ask for permission to participate. The client may allow a team member to serve as a volunteer employee for a couple of hours. At GB Video, the team members for a small outlay of cash certainly can participate as rental customers and may gain good insight by so doing.

The team wants to be proactive in collecting information and developing insight. The team easily can lose out on a good source of information simply because the team neglected to ask. However, if the client denies a request, good professionals accept the decision with grace and understanding.

CURRENT SITUATION NARRATIVE MODEL

The current situation **narrative model** serves two purposes: (1) to describe the relevant aspects of the current situation in a representation that most clients understand and (2) to provide cross-checks for the completeness and correctness of the graphical models. The narrative describes the issues and problems from the current situation that are relevant to the proposed system design in a set of well-written and well-organized paragraphs. The narrative always should use terms and concepts that the client understands. The client normally will understand the terms that describe the operations of the organization but may not understand such terms as *DFD* unless the client comes from an IT area. While each project will differ, some general guidelines will help in the construction of the narrative. The narrative contains four sections:

1. The description of current operations.
2. The description of physical and organizational infrastructure.
3. An analysis of problems in the current operations.
4. An analysis of the aspects of the current situation that the proposed system retains and the aspects that change.

Each of the sections is described below.

Description of Current Operations

The description of **current operations** describes the activities that take place in an organizational context: who, what, where, and when. When possible, the description should read as a story or mini-play. Often the narrative makes most sense when it follows the natural sequence of the activity. For example, in a video store the first process is “Rental” and the second is “Return.” But, the process “New Member” may precede everything because only members may rent tapes and DVDs. The narrative should describe fully the operation and read well, that is, be complete, correct, and follow the rules for good writing. A set of cryptic lists and disconnected sentences do not make a narrative model.

Much of the time, the activities that the proposed system will perform currently take place in some form. For example, a new Web site may provide information that people currently get from less current, complete, and convenient other sources. In this case, the current operation narrative will describe the what, where, who, and how for the way the information currently is provided.

The content for the description of the narrative for current operations should follow the concepts from data and process analysis. The narrative may include text descriptions for the following:

- **Events.** The event(s) that initiate action. In some batch processing operations, for example, preparing monthly management reports or utility bills, the only event is reaching a particular time. More interactive operations—video rentals, catalog sales, Web pages—may encompass a number of trigger events. In GB Video the trigger events include when (1) a customer asks to become a member; (2) a customer asks to rent one or more videos; and (3) a customer returns a video.
- **Data flows and processes.** Each trigger event directly or indirectly causes data flows through the various processes to the outcome events. In GB Video, a rental request involves input data from externals: video IDs, customer number, credit card number, and expiration date. Tracing the data flows shows that these data flow through a number of processes to create a invoice record, compute the rental charge, collect a payment from the customer, retrieve data from the video rental and customer records, and prepare a receipt for the customer. The receipt is an outcome event, which is an action or data set that leaves the system. Data collection, storage, and retrieval systems may contain only a few processes to input and store new data and retrieve data. Transaction processing systems may contain a large number of processes.
- **Data content.** Each data flow and data store will contain data about such “things,” as a customer, or a video. For management reporting or query activities or for some Web site operations, identification of the data structure may comprise the bulk of the analysis task.
- **Other issues.** The description also should address any important issues of timing, frequencies or volumes, file and record sizes, and special business concerns.

While collecting information to prepare the narrative, the team may encounter forms, reports, and other materials. Copies of all relevant materials should appear

either in the narrative or in an appendix. All documents included always are referenced or cited in the text of the narrative and explained as appropriate. When the team studies a current system, much of the aforementioned information on events, data, and process may come from the system documentation. If no or only incomplete system documentation is available, the team may need to *reverse engineer* the system, to infer the data and process structure or models by analyzing the system inputs, outputs, and other materials.

An example of a description of current operations for GB Video appears in the narrative model in Figure 7.1.

Physical and Organizational Infrastructure

In most circumstances, the current situation will contain a **physical infrastructure**, including I/O devices, data storage devices, processors, telecommunications, operating systems, and programs; and an **organizational infrastructure**, including people, roles, authority, and responsibility. Even a manual system may contain telephones, files, calculators, and such. This section of the narrative defines and describes the infrastructure. The description should cover only the infrastructure relevant to the project statement. For the GB Video project, stores, videos, files, forms, shelves, return bins, and cash registers probably are relevant but Mr. Crosier's office, the check printing machine, and the employee lounge probably are not. The organizational structure for clerks making rentals and returns is relevant; the organizational structure for payroll probably is not. The organizational and physical infrastructures may have sketches, charts, or pictorial models that specify components and show how they are connected, or the team may wish to prepare such a model. Figure 7.1 shows a description of infrastructure for GB Video.

Problem Analysis

As noted previously, the client often wants a proposed system to alleviate problems that the client believes exist in the current situation. Even without prompting, the client often will identify a number of problems. Typical problems are that the system is too slow, contains errors, costs too much, stores redundant data, is disliked or not used by users and/or customers, doesn't supply the information or functions that the client wants. Alleviating the problems should improve the performance measures associated with the objectives the team identified during the strategic alignment analysis. Normally the team describes the problems as clearly as possible in text form. Identification of as many of the perceived problems as possible is important because the perceived problems suggest possible alternatives and features for the proposed system. A sample **problem analysis** for GB Video appears in Figure 7.1.

Retention and Change Analysis

The **retention and change analysis** of the current situation defines the parts or aspects of the current situation that (1) should remain in the proposed system and (2) may change. A proposed system that omits one or more critical features of the current system may force the client to run both systems until the new one includes the additional capability. To develop the analysis, the team reviews the description

FIGURE 7.1 Current Situation Narrative Model for GB Video

Current Situation Narrative Model

Introduction

The narrative model for the GB Video Rental and Return system contains sections that cover a description of the current operations and the physical and organizational infrastructure, an analysis of problems in the current operations, and an analysis of the aspects of the current situation that the proposed system retains and the aspects that change. A discussion of activities the team plans to undertake and the client deliverables appears in the statement of work in Appendix A.

Description of Current Operations

In consonance with the request of Mr. Cosier, the team looked at video rental and return and closely related activities. GB operates in a manner similar to most video stores. GB rents only to customers who are members. If a customer wishes to become a member, GB will issue a membership provided the customer has a credit card, telephone, and a government-issued picture ID (driver's license, etc.). The GB clerk obtains the name, address, credit card number, and expiration date from the picture ID and credit card; asks the customer for a telephone number; prints this data on a customer form and assigns a unique member number. The clerk prepares a membership card with name and member number and gives it to the customer. The customer form is placed in the customer file box. On every contact, the clerk asks the customer about changes and updates the customer form any time a customer reports a change. Exhibit 1 in Appendix B shows a customer form.

Although most staff members at GB Video talk about renting "videotapes," GB actually rents more DVDs than tapes. This report uses the term *video* to refer to both tapes and DVDs. Customers may rent one or more videos for one or more days. The customer finds the desired videos and brings them to the counter. The clerk copies the name and number from the customer card and copies the title and unique ID number from the label on each video onto a prenumbered invoice form. Exhibit 2 in Appendix B shows an invoice form. If the customer forgets his/her member card, the clerk looks up the number in the customer file. The clerk also enters his/her employee number, the date of rental, and a due date for each video; computes the amount of the charge and tax; and gives a copy of the invoice to the customer as a receipt. Customers may pay for rentals with cash, check, or a credit card. The clerk notes the payment type on the invoice. For credit card payments, the clerk runs the customer's credit card through the credit card terminal and keys in the amount. The credit card company approves or denies the charge.

As time permits, a back office clerk processes the invoices. The date and rental number for each rental are entered on the file card for the video in the video rental file. The header data on the card for each video owned by the store shows the video number, title, vendor number, and date acquired. Purchasing enters this header data when a video is received. In this manner, the store can track the status of each video. Exhibit 3 in Appendix B shows a video rental card.

When a video is returned, a back office clerk retrieves the card for the video rental and the invoice, records the return date on the invoice and the video rental card, calculates overdue charges if any, and processes the credit card transaction. The credit card number is obtained by retrieving the customer form from the customer file. The overdue charges are noted on the invoice. The videos from a rental may be returned on different dates.

Once a week, the office uses the copies of the invoices and an employee file to prepare a report for the store manager showing the number of rentals for each video and for each clerk. The completed invoices and copies of video rental forms for videos that are more than three days overdue are sent to Accounting. As part of a separate system, the accounting clerk uses the invoices to get revenue data for accounting records and sends out overdue notices to members as required.

The office also uses the video rental cards and the vendor file to prepare and mail a monthly report to each vendor showing the rentals for each video supplied by the vendor grouped by title.

Graphical data and process models for the current operation appear in Appendix C.

Physical and Organizational Infrastructure

As noted in the description of current operations, GB Video operates a mostly manual system. The system uses paper forms and index cards in file boxes for data input, output, and storage. The clerks write on the forms and cards with pencil or pen. The credit card terminal is the only electronic communication device in the system.

Each store owns and manages its own data. The store manager controls the operation of the store information system. The clerks and all other employees in the store report to the manager. Clerks are allowed to create and update invoice and customer records. Video rental records are created by Purchasing and updated by the clerks.

Problem Analysis

Mr. Cosier and his staff identified the following problems in the current situation:

1. Because of the manual system, GB requires more clerks than similar stores with automated systems resulting in a higher cost and longer elapsed time per transaction.
2. Customers complain about long lines and slow checkout of rentals. Some frustrated customers dump their videos on the counter and go off without completing the rental.
3. Delays occur when several clerks wish to use the member or video card file at the same time.
4. Mistakes are common. For example, the video card may indicate that the video is on the shelf because it shows a return date and no new rental date, but the actual video cannot be found. Clerks make mistakes in computing the charges. Customers often correct overcharges but remain silent when the clerk undercharges.

5. People, including some members, use false member numbers and names to rent videos. If the customer does not show a member card, the clerk is supposed to check the member file. Because of the time and effort needed to check the file, the clerks omit checks most of the time, especially when the store is busy. GB experiences a higher nonreturn or loss rate for videos than other similar stores.
6. Accounting gets busy and does not send overdue notices to customers on a timely basis. Sometimes a video is several weeks' overdue before the customer receives a notice. Customers use the late notice as a reason to refuse to pay overdue charges.
7. Because of the expense of preparing a manual mailing, GB does not do any direct mail marketing to members. Other stores have successfully used direct mail marketing to increase revenues.
8. The rental data summarized by video is costly to prepare and of questionable accuracy. No data exist on customer preferences. Purchasing often uses guesses, estimates, and periodic direct observation to determine which videos and how many copies to buy.

Mr. Cosier estimates that correcting these problems should increase revenue by at least 5 percent and at the same time reduce costs by over \$55,000 a year.

Retention and Change Analysis

The proposed system will retain, at the conceptual level, the functions in the current system to enroll a member, rent a video, and return a video. The proposed system will continue to collect all of the data collected by the current operation. However, the data model will change to one based on the "things" about which data are collected rather than on the forms and file cards in use. The proposed system may contain additional functionality to prevent nonmembers from renting videos.

The reporting functions will move to another system. The Rental and Return System will make the data it collects and stores available to the other systems. As a result of the move of the reporting functions and the new data model, a number of the data flows will change. Since the purchasing, reporting, and accounting functions will access the rental return data with their own systems, the Rental and Return system will show no flows to Accounting, Management, and Vendor. The contents of flows to and from the Invoices and Video Rentals stores will change to match the new data model. The flows to and from the externals Customer and Credit Card Company remain unchanged in content.

The physical infrastructure will change from one based on manual operations to a computer-based infrastructure. The proposed system will have computer input and output devices that do not exist in the current system. The organization infrastructure will change in one respect—the clerks will interact with computer I/O devices.

In this case, the analysis of the current situation provides a good base upon which to design the proposed system. With the structure outlined above, GB Video should realize their strategic goals of increases in profits and sales.

Appendix A. Statement of Work. (Not included here, see Chapter 3.)

Appendix B. Forms and Documents. This appendix contains forms used by GB Video in the current Rental and Return system. The role and use of each form is described in the narrative. The documents in this appendix are

- Exhibit 1. Member Data Card
- Exhibit 2. Invoice and Customer Receipt
- Exhibit 3. Video Rental Card

Exhibit 1. Member Data Card

GB Customer Record	
Richard Jazzperson	1346
Name	Member Number
307 Brooks Norman, Oklahoma 73019	VISA 9444 5432 6666 1234 04 09
(405) 325-0768	

Exhibit 2. Invoice (Copy 1) and Customer Receipt (Copy 2)

GB Video Stores					
Date:	11-2-2006	Emp # :	175	RENTAL NO.	1715
Member :	Richard Jazzperson	Member No :	1346		
Video #	Title	Due Date	Cost	Return Date	Overdue Charge
15751378	Patriot Games	11-3	\$ 2.00		
6613498	African Queen	11-4	\$ 3.00		
			\$		
Pay Type : Cash	<input checked="" type="checkbox"/>			Tax :	\$.40
Credit	<input type="checkbox"/>			Total :	\$ 5.40
THANK YOU				Copy 1 Store	

Exhibit 3. Video Rental Card

Video No :	6613489	Title :	African Queen
Date Acquired	6-1-06	Vendor :	129

Rental No.	Date Out	Date In	Rental No.	Date Out	Date In
1497	6-3-06	6-4			
1558	8-7-06	8-9			
1579	8-20-06	8-21			
1715	11-2-06				

Appendix C. Graphical Data and Process Models

This appendix contains the data and process models for the GB Video System. The models included in the appendix consist of:

- Exhibit 1. Context Level DFD for GB Video Current Operation
- Exhibit 2. First Explosion DFD for GB Video Current Operation
- Exhibit 3. Metadata for the GB Video Current Operation DFD
- Exhibit 4. Enterprise Data Model for GB Video

(The models are not included here. The figures appear later in this chapter.)

Page 5

of the current operations, the problem analysis, and the proposed system features described in Chapter 6.

An example of a retention and change analysis appears in Figure 7.1. An overall structure of paragraphs reads more easily than lists, but the analysis may contain lists and/or tables within the paragraphs as appropriate.

Correctness and Completeness with Multiple Representations

While the narrative model should describe fully the current situation, more structured models can help team members to identify errors and omissions and to communicate with each other and the client. Typically a team will select DFDs as the graphical process model and an ERD as the graphical data model. The **multiple representations**, the narrative and **graphical models**, must match. The team's graphical data and process models should provide **consistency**. Pictures of the current operation should include *all of and only* the data and processes that appear in the narrative model for the current operation. If the narrative describes

data or processes that do not appear on the EDM and/or DFDs, then the diagrams and/or the narrative must contain errors or omissions and lack correctness. If the DFDs and/or EDM contain items that do not appear in the narrative, again one or more errors must exist and the diagrams lack **completeness**. In systems work, completeness and correctness are critical properties.

CURRENT OPERATION GRAPHICAL PROCESS MODEL

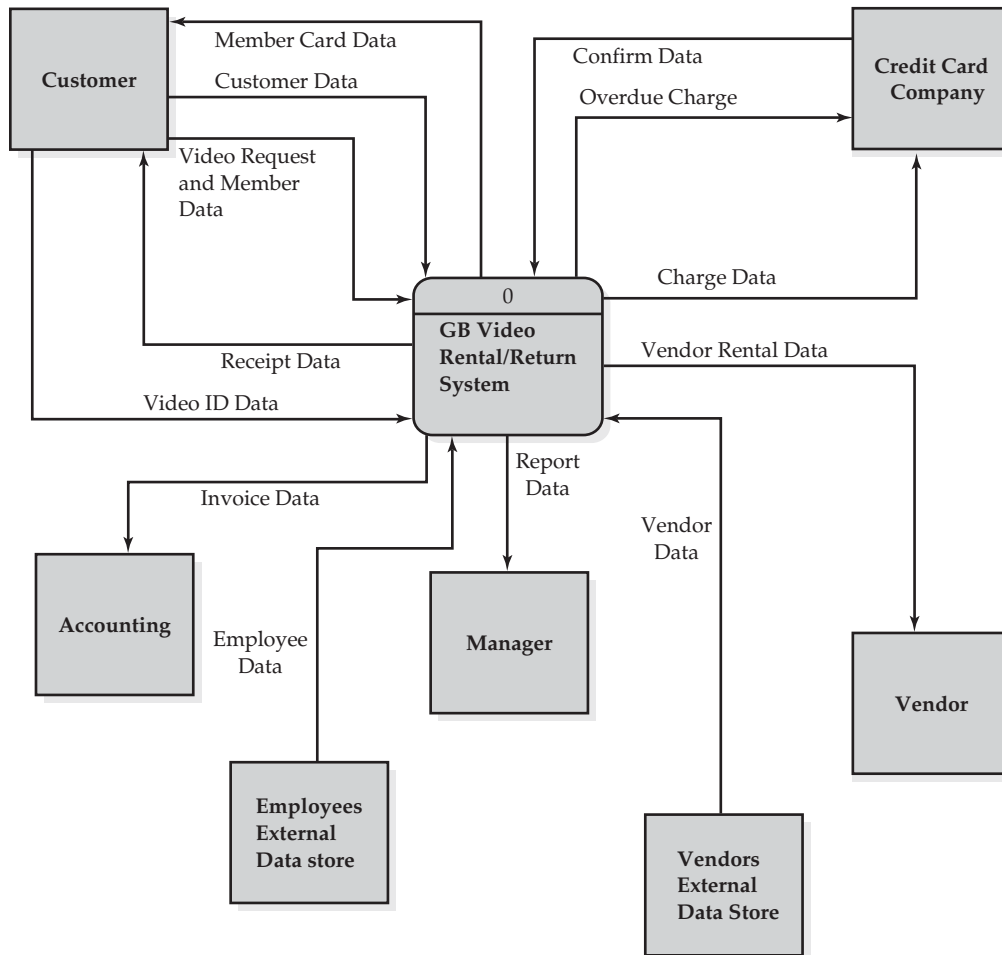
While the team can choose from a number of **process models**, in many cases DFDs provide a good representation for understanding the current operation. Such representations as object-oriented models, process hierarchy diagrams, and page maps, often contribute more when used as part of the proposed system design. Data flow diagrams for the current operation provide a high-level model of the physically existing data flows, stores, and processes. DFDs present the data and process information from the narrative model in a pictorial or graphical format. The narrative should contain a clear and explicit reference to the associated DFDs. If the client has an IT background, the DFDs may appear as figures in the text body; otherwise, DFDs probably fit best in an appendix. If the narrative already exists, the team can identify the major processes, the data flows, the stores, and the externals in the narrative and translate them to the DFD.

Guidelines

Chapter 5 covers the concepts and mechanics of data flow diagrams. A reader unfamiliar with DFDs should read or review Chapter 5 prior to reading this section. In many cases, graphical models of the current operation provide an overview and general insight. As noted previously, the current operation models are not intended as detailed guidance for programmers or vendors; the proposed system models in Chapters 8 and 11 fill this role.

The guidelines for creating current operation DFDs include the following:

- *DFDs should follow the rules described in Chapter 5.*
- *Every DFD should be referenced in the narrative.*
- *A context-level DFD can provide a high-level overview of how externals interact with the system in the current operation. The externals and flows on the diagram should match the description in the narrative.*
- *A first explosion DFD can identify the main processes, data flows, and stores. Normally a first explosion DFD provides all the detail about the current system that is helpful for proposed system design. When a project involves automating a manual system or changing the physical infrastructure for an existing system, the current operation process model also may serve as the proposed system process model and thus may involve substantially more detail and depth.*
 - *The first explosion process boxes should match the narrative and provide enough detail to give a clear idea of what is happening. For most projects, two processes probably are too few; 20 probably are too many. Some of the processes in the current operation may lie outside the boundary of the proposed system.*

FIGURE 7.2 Context-Level DFD for GB Video Current Operation

- Data stores that are owned and/or maintained by other systems can appear as externals outside the system boundary on the DFDs.
- *The metadata provide a brief description of every object on the DFDs—external, store, flow, and process.*
- *After creating each draft of the DFDs, the team should check carefully to assure that the DFDs and the narrative model are consistent, complete, and correct.*

The GB Video context-level DFD is shown in Figure 7.2 and the first explosion DFD in Figure 7.3. In Figures 7.2 and 7.3, the DFDs correspond directly to the current operation description in the narrative model. Every process in the narrative appears on the DFD (unless it is described as part of a separate system), and every process on the DFD appears in the narrative. The dotted line in Figure 7.3 identifies the boundary of the system; everything inside the dotted line box on

FIGURE 7.3 First Explosion DFD for GB Video Current Operation

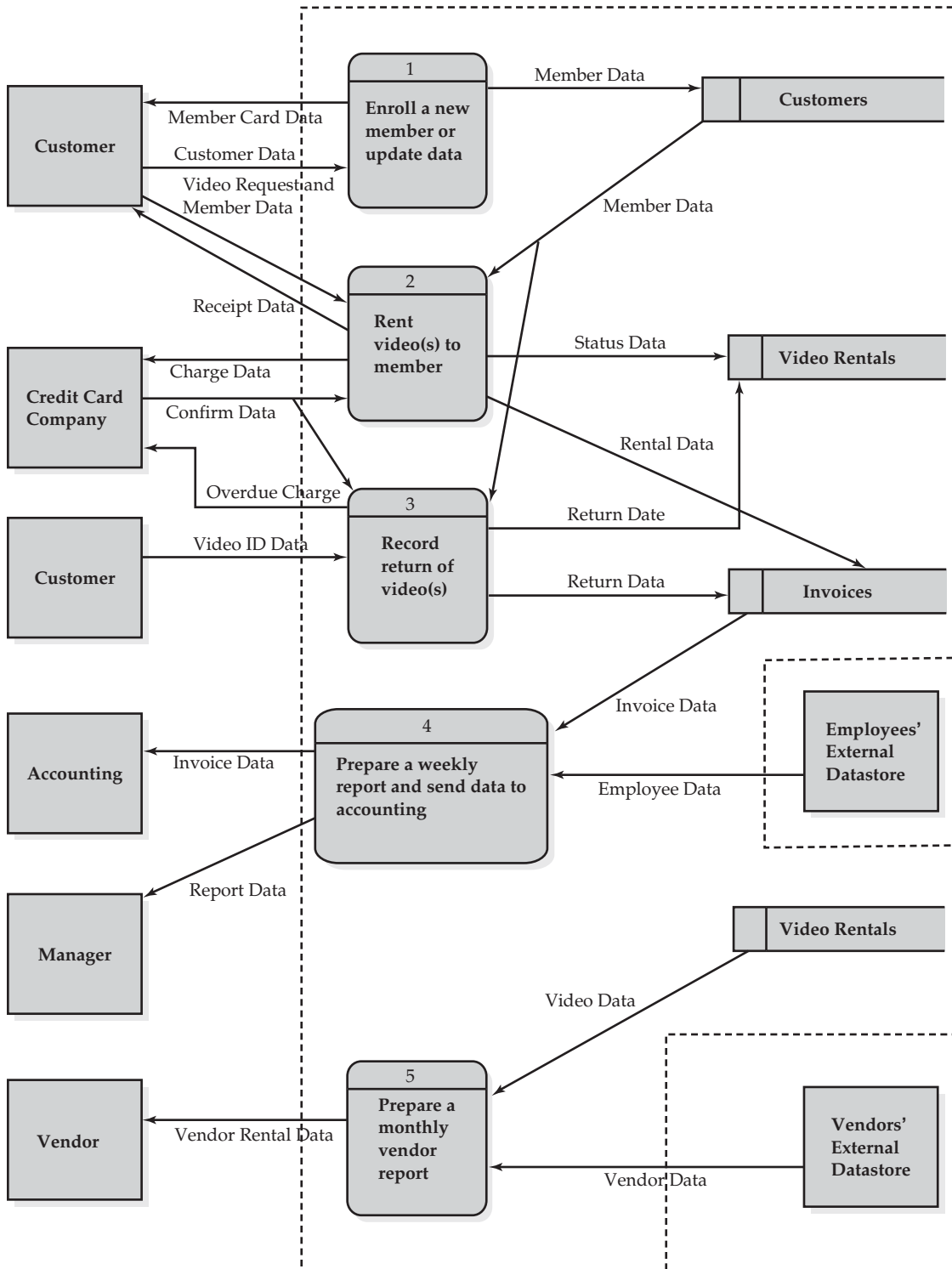


Figure 7.3 lies inside the system process box on Figure 7.2. Some of the processes in the DFD, for example, processes 4 and 5, will lie outside the boundary of the new system but are included because they are part of (inside the boundary of) the current operation. The DFD in Figure 7.3 follows the rules for DFD explosions found in Chapter 5. In this example and in many current operations some existing physical data stores modeled in the DFD may contain attributes from several entities. For example, the data store Customers contains only the attributes of the entity Customer, but the data store Invoices contains attributes of the Customer, Rental, Employee, and Video entities.

The DFD in Figure 7.3 assumes that the Video Rentals data store is owned by the Rental and Return System and thus belongs inside the system boundary. The rental/return system probably owns the Invoices and Members data stores so these data stores also appear inside the system boundary. The data stores called Vendors and Employees receive no input in the DFD. This lack of input reflects the fact that they are owned by different systems, and thus they appear as externals outside the system boundary on the DFD. The correct identification of ownership adds value to the DFD. When an analyst sees a data store outside the system boundary on a DFD, the analyst knows immediately about the requirement to contact and work with the owner of the data store on access and changes.

Process Model Metadata

Metadata, or data about data, describe the relevant features of each of the objects that appear in the graphical models of the system. A viewer with a question about the meaning or significance of some portion of the graphical models can refer to the metadata for more information. For the current operation process model, the metadata may consist of short, concise text descriptions for each object—external, data flow, process, and data store—on the DFD. The metadata should identify clearly the content of data flows and data stores. The process descriptions should come directly from the text of the current operation narrative, rearranged or reformatted as appropriate. All of the metadata descriptions should match the content of the narrative. Sample metadata for the GB Video Current Operation DFD appear in Figure 7.4.

CURRENT OPERATION GRAPHICAL DATA MODEL

A **data model** provides a graphical tool to define the underlying conceptual data structure for the current operation. The data model can help the team to understand the content of the current operation and to check for correctness and completeness with the narrative model. The team can use an ERD model to graphically represent the underlying conceptual data structure. The physical data stores in the current system may contain data from several entities. The graphical data model should be consistent with both the narrative model and process model for the current situation.

Because the goal is to provide a broad overview of the current operation in preparation for the design of the proposed system, the team may wish to use

FIGURE 7.4 Metadata for the GB Video Current Operation DFD

Externals

Accounting. The person who keeps the financial and accounting data at GB Video.
Credit Card Company. The company that processes credit card transactions for GB Video.
Customer. People who wish to rent videos at a GB store. A customer must become a member to rent a video.
Manager. GB Video store manager(s) and Mr. Cosier.
Vendor. Companies that sell videos to GB.

Data Flows

Charge Data. Credit card number, expire date, and transaction amount.
Confirm Data. Either an approval or a reject message.
Customer Data. Name, address, telephone and credit card info for a potential member supplied by a customer to become a member or to update his/her record.
Employee Data. The employee's number.
Invoice Data. All of the data on completed invoice forms.
Member Card Data. Name and member number printed on a member card.
Member Data. Customer data plus a member number.
Overdue Charge. Credit card number, expire date, and transaction amount.
Receipt Data. The data on the receipt given to the customer including rental number, employee number, customer number, name, rental date, title, planned return date, payment type, cost, tax, and total charge.
Rental Data. The member, video, and rental transaction data.
Report Data. An analysis of the data on completed invoice forms.
Return Data. The date each video is returned plus the overdue charge if any.
Return Date. The date the video is returned.
Status Data. The rental number and rental date.
Vendor Data. The vendor numbers, names, and addresses.
Vendor Rental Data. The analyzed data about rentals from video rental cards.
Video Request and Member Data. The numbers of the videos that the customer wishes to rent plus the customer's member number or other identifier.
Video Data. The video number, vendor number, and rental dates for each video.
Video ID Data. The video number of a returned video.

Data Stores

Customers. Contains customer data on paper forms arranged alphabetically. A copy of the form appears in Exhibit 1.
Employees. Contains personal and payroll data for employees at GB Video. Each employee has an employee number.
Invoices. Contains rental, customer, employee, and video data on paper forms, one for each rental arranged by rental date. A copy of the form appears in Exhibit 2.
Vendors. Contains in a notebook an alphabetical list of names with addresses for vendors.
Video Rentals. Contains video and rental data on index cards, one for each rental video arranged alphabetically by title and then by video number within a title. A copy of the card appears in Exhibit 3.

Processes

C1.0 Enroll a new member or update data. GB operates in a manner similar to most video stores. GB rents only to customers who are members. If a customer wishes to become a member, GB will issue a membership provided the customer has a credit card, telephone, and a government-issued picture ID (driver's license, etc.). The GB clerk obtains the name, address, credit card number, and expiration date from the picture ID and credit card, asks the customer for a telephone number, prints this data on a customer form, and assigns a unique member number. The clerk prepares a membership card with name and member number and gives it to the customer. The customer form is placed in the customer file box. On every contact, the clerk asks the customer about changes and updates the customer form any time a customer reports a change.

C2.0 Rent video(s) to member. Customers may rent one or more videos for one or more days. The customer finds the desired videos and brings them to the counter. The clerk copies the name and number from the customer card and copies the title and unique ID number from the label on each video onto a prenumbered invoice form. Exhibit 2 in Appendix B shows an invoice form. If the customer forgets his/her member card, the clerk looks up the number in the customer file. The clerk also enters his/her employee number, the date of rental and a due date for each video, computes the amount of the charge and tax, and gives a copy of the invoice to the customer as a receipt. Customers may pay cash or use a credit card for rentals. The clerk notes the payment type on the invoice. For credit card payments, the clerk runs the customer's credit card through the credit card terminal and keys in the amount. The credit card company approves or denies the charge. As time permits, a back office clerk processes the invoices. The date and rental number for each rental are entered on the file card for the video in the video rental file. (The header data on the card for each video owned by the store shows the video number, title, vendor number, and date acquired. Purchasing enters this header data when a video is received.) In this manner, the store can track the status of each video.

C3.0 Record return of video(s). When a video is returned, a back office clerk retrieves the card for the video rental and the invoice, records the return date on the invoice and the video rental card, calculates overdue charges if any, and processes the credit card transaction. The credit card number is obtained by retrieving the customer form from the customer file. The overdue charges are noted on the invoice. The videos from a rental may be returned on different dates.

C4.0 Prepare a weekly report and send data to Accounting. Once a week, the office uses the invoices and an employee file to prepare a report for the store manager showing the number of rentals for each video and for each clerk. The completed invoice and copies of video rental cards for videos that are more than three days overdue are sent to Accounting. As part of a separate system, the accounting clerk uses the invoices to get revenue data for accounting records and sends out overdue notices to members as required.

C5.0 Prepare a monthly vendor report. The office also uses the video rental cards and the vendor file to prepare and mail a monthly report to each vendor showing the rentals for each video supplied by the vendor grouped by title.

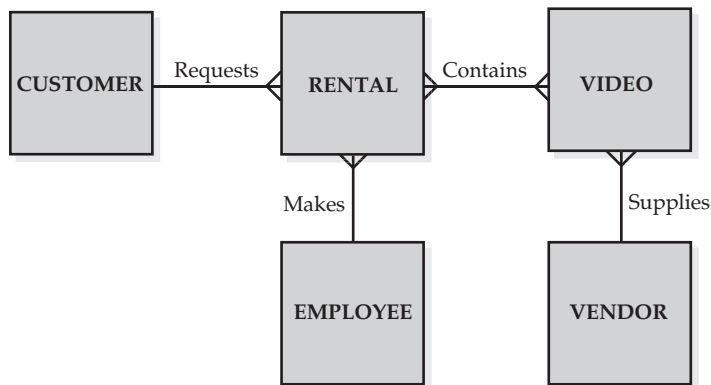
such a high-level ERD as an enterprise data model (EDM) to identify the entities and relationships that correspond to the data in the data stores shown on the DFD for the current operation. Information on the attributes of each entity often is unnecessary for the current situation. In the project definition report, the EDM can appear as a figure within the text if the client has an IT background; otherwise, the EDM goes in an appendix. The EDM and the metadata for the EDM should correspond to the descriptions in the narrative and to the DFDs. The narrative must contain a clear and explicit reference to the associated EDM. The EDM or any other ERDs should follow the rules listed in Chapter 4.

The EDM in Figure 7.5 shows the underlying entities of Customer, Rental, Employee, Vendor, and Video as described in the narrative and exhibits for the GB Video rental and return current operation. Figure 7.5 also contains metadata for each of the entities. For the current system, a simple text description provides adequate metadata.

Some of the data stores, forms, and reports associated with the current operation contain data from several different entities. For example, the invoice in the Invoices data store and the receipt that a GB Video customer receives to document a rental contain attributes from the entities of Customer—name, address, and other customer information; Rental—date, rental number; Employee—employee number; and Video—video number title, description.

As noted, the current Rental system does not manage the data associated the Employee and Vendor entities. However, data from the Employee and Vendor entities are part of the current operation. An employee makes every rental, and

FIGURE 7.5
Enterprise
Data Model
for GB Video



Entity Metadata

CUSTOMER. A customer is a person who rents videos from GB Video. Before a customer can rent a video, the customer must become a member. A customer may make multiple rentals.

RENTAL. A rental is the set of transactions for a customer to rent and return one or more videos.

VIDEO. A video is a tape or DVD that may be rented to multiple customers.

EMPLOYEE. An employee is a clerk, manager, or other person employed by GB Video who may handle multiple rentals.

VENDOR. A vendor is a firm that may sell multiple videos to GB Video.

the employee number appears in the current invoice. In similar fashion, vendors supply the videos and a vendor number appears in the current Video Rental record. The current report preparation processes also use data from both Employee and Vendor entities. The EDM includes Vendor and Employee to correctly identify all of the entities associated with the current operation. The analyst can and should exercise some discretion on what entities to include in the EDM. The goal is clarity with respect to what to retain and change in the proposed system.

THE PROJECT DEFINITION PRESENTATION

At a number of points during the project, the team may seek client approval to proceed. The client and team agree on the specific review points for each project. Typical review points include asking for client approval of the set of activities the team will undertake and the deliverables (the statement of work), the strategic alignment, and the proposed system features, constraints, and procurement options (the project definition presentation), and the recommended solution (often a meeting with the client). Some clients will ask for more approval points. Each weekly or other progress report provides the client the option to approve continuing the project. The client might decide to cancel or end the project at any of the review points, but more often the client either approves or suggests changes.

When the team has collected enough information to prepare drafts for the problem definition report, the team can proceed with a formal project definition presentation for the client. The overall goal of this meeting is to assure to the extent possible that the team and client visualize the same requirements for the proposed system. Possible objectives of the presentation are to achieve

- **Correctness.** Get the client to interact with the team members to correct any errors or omissions and/or to provide additional information.
- **Competency.** Demonstrate to the client that the team can construct an adequate overall view of the current situation and strategic alignment and features, constraints, and alternatives for the proposed system.
- **Feedback.** Show the client that the team understands and listens to the client.
- **Approval.** Give the client the opportunity to approve or disapprove of the team's work to date and to take any actions the client deems appropriate.

Normally a project definition presentation runs for about an hour. The team should prepare an agenda for the meeting and should share it with the client in advance, if possible. The typical order of the presentation is

1. Introductions.
2. Thank the clients for attending and ask permission to record if needed.
3. Review the agenda.

4. Summarize the materials from the team's draft Problem Definition Report.
 - a. Problem statement.
 - b. Current situation overview.
 - c. Strategic alignment.
 - d. Features and constraints for the proposed system including procurement options.
5. Review the team's draft statement of work making sure the client engages and comments.
6. Summarize the key points stressing any ideas, changes, corrections, and other comments received from the client.
7. Thank the clients.
8. Close the meeting.

The team should tailor the presentation to the clients. Charts with lists, graphs, diagrams, and other devices can help the client to understand rapidly what the team plans for the content of each of the sections. The team should make sure that the client can read all of the content of the visual aids from the expected viewing distance. Flip charts, charts drawn on a whiteboard, and such are easy to change and encourage discussion. Computer-generated charts are a little more difficult to change during the discussion but can work well provided that the team takes steps to generate discussion during the presentation. Diagrams and charts tend to stimulate discussion better than lists. If the clients are not IT specialists, the presentation should avoid such IT terms as ERD, and DFD. Although the team may wish to show an ERD chart, team members should describe it as a chart that shows the data in the system not as an ERD.

The "correctness" objective is achieved only by getting the clients to comment on corrections and additions. The team should strive for a highly interactive session. At the beginning of the review, the team can invite the clients to comment as the presentation proceeds. Sometimes the clients will make comments without further urging. The team can encourage the clients to interrupt by stopping the team presentation and letting the clients talk at the first sign of any comments or concerns. After each part of the presentation, the team should stop and ask for additional comments. If the client offers no comments, the team can prepare and ask the client questions. The team can demonstrate to the client that the team hears the comments by marking changes or suggestions on the charts or by repeating the client's key points.

While the team wants to encourage client interaction, the team nonetheless should control the flow of the meeting. If the client goes off into personal stories or other nonrelevant material, the team should listen politely and not interrupt. But when the client stops talking for a moment, the team can use the agenda to get the discussion back on track. For example, a team member can say, "Shall we move on to the next topic?" When time or topics of conversation run out or the client wants to leave, the team should end the meeting. The team can handle any unresolved items by telephone or e-mail or at another visit. Additional material on presentations appears in Chapter 3.

COMPLETING THE PROJECT DEFINITION STAGE

Once the team makes the Project Definition Presentation, the team can complete the Project Definition Report. The team should enter any corrections or changes that arise during the presentation into the report and the Statement of Work (SOW) as appropriate. When the team has a final corrected copy of the SOW, the team asks the client to sign the SOW to get a written expression of the client's approval to proceed with the project.

Toward the end of problem definition, the team can update the project plan for the project. As the team carries out the problem definition activities, the team will learn more about what the client wants, the current situation, and the activities appropriate to the next stages of the plan. The team can use this information to prepare or revise the detailed steps for the Proposed System Phase and to make other revisions as needed. In all cases, the team should make sure that the client and manager understand and agree with any revisions that affect them. A revised project plan for GB Video appears in Table 7.1.

TABLE 7.1 Revised Project Plan as of February 12

Milestone or Activity	Est. Per. hours	People Assigned	Sequence PreReqs	Start Date	Draft Date	Due Date	Status
1. Organization Plan				13 Jan	19 Jan	21 Jan	Complete
1.1 Deliverables							
1.1.1 Team contract	10	All		13 Jan	19 Jan	21 Jan	Handed in
1.1.2 Skill inventory	5	All		13 Jan	17 Jan	21 Jan	Handed in
1.2 Meet with manager	6	Terrie/Dan	1.1	19 Jan	20 Jan	21 Jan	Done
1.3 Meet with client	18	All	1.2	13 Jan	23 Jan	24 Jan	Done
2. Project Definition							Complete
2.1 Review client request				24 Jan	31 Jan	31 Jan	Done
2.2 Draft SOW	12	All	1.3	24 Jan	24 Jan	24 Jan	Done
2.3 Draft project def. rpt.	40	Dan/Al	2.1	24 Jan	31 Jan	13 Feb	Draft done
2.4 Proj. def. presentation	23	Dick/Al/Dan	1.3	24 Jan	3 Feb	6 Feb	Done
3. Proposed System				1 Feb	28 Feb	28 Feb	Started
3.1 Prepare specifications	20	Al/Terrie	2.3	1 Feb	20 Feb	20 Feb	Started
3.2 Draw models				1 Feb	20 Feb	20 Feb	Started
3.2.1 Data model	5	Al	2.3	1 Feb	20 Feb	20 Feb	Done
3.2.2 Process model	20	Dan	2.3	1 Feb	20 Feb	20 Feb	Started
3.3 Refine alternatives	5	Dick/Terrie	2.3	5 Feb	12 Feb	12 Feb	Done
3.4 Conduct evaluation	15	Dick	3.3	13 Feb	20 Feb	20 Feb	Not started
3.5 Select recommendation	2	All	3.4	20 Feb	28 Feb	28 Feb	Not started
4. System Acquisition			3.5	1 Mar	15 Mar	15 Mar	Not started
Option A. Buy							
Option B. Build							
5. System Delivery				15 Mar	8 Apr	8 Apr	Not started
5.1 Acceptance test							Not started
5.2 Implementation							Not started
6. Final Report and Presentation				5 Mar	18 Apr	23 Apr	Not started

Summary

The goal of current situation analysis is to obtain information to answer the following questions:

- What happens in the current operation? What and how do events, processes, and data interact as the current operations start, proceed, and end?
- What existing physical and organization infrastructures support the current operations?
- What problems result from the current operations and infrastructure? What things happen or do not happen that the client considers undesirable?
- How do the problems in the current situation affect the performance measures identified during the strategic analysis?
- What aspects of the current situation may or must appear in the proposed system solution? What operations, problems, and infrastructure carry over to the proposed system?
- What aspects of the current situation should change in the proposed system solution? How should operations, problems, and infrastructure change?

The level and extent of the analysis will depend on the nature of the project, the background information possessed by the team, and the desires of the client and the project manager. The team might prepare a narrative model, enterprise data model, and data flow diagram for the current operation. Graphical and narrative representations help establish completeness, correctness, and consistency, and the representations provide a convenient way to organize and cross-check data about current operations.

Narrative models describe the “who, what, where, and when” of the current situation in well-written paragraphs. Narratives should identify events, data inputs from externals, the outputs to externals, internal data flows, data stores, data flows to store and retrieve data and processes. A section of the narrative lists the organizational and physical infrastructure for the current operation. The team also conducts a problem analysis to identify the problems that exist in the current situation and a retention and change analysis.

More structured models can help team members to identify errors and omissions and to communicate with each other and the client. A team might select DFDs as the graphical process model and an EDM as the graphical data model. The team’s graphical data and process models should provide a coherent and complete picture of the current operation that includes all of and only data and processes that appear in the narrative model for the current operation. In systems work, completeness and correctness are critical properties.

The current operation DFDs model the externals, processes, flows, and stores as they physically exist in the current operation. The existing data stores may contain data from several entities. The enterprise data model presents a graphical overview of the conceptual data structure for the current operation. The EDM may include entities that relate to the current operation but will lie outside the boundary of the proposed system.

At a number of points during the project, the team may seek client approval to proceed. The client and team agree on the specific review points for each

project. Once the team completes a draft of the project definition materials, the team may conduct a project definition presentation with the client in order to (1) obtain client approval to proceed with the project; (2) provide feedback—show the client that the team understands and has listened to what the client said; (3) demonstrate competency—demonstrate to the client that the team can construct an adequate overall view of organizational strategy, goals and features for the proposed system, constraints, alternatives, and the current situation; and (4) correct the project definition materials—get the client to interact with the team members to correct any errors or omissions and/or to provide additional information. The team uses the insight gained from the presentation to create a final project definition report including a statement of work.

As the team carries out the problem definition activities, the team will learn more about what the client wants, the current situation, and the activities appropriate to the next stages of the plan. The team can use this information to prepare or revise the detailed steps for the Proposed System phase and to make other revisions as needed to the project plan.

Key Terms

approval, 247	data model, 243	organizational
competency, 247	feedback, 247	infrastructure, 234
completeness, 240	graphical model, 239	physical infrastructure, 234
consistency, 239	metadata, 243	problem analysis, 234
correctness, 247	multiple	process model, 240
current situation, 229	representations, 239	retention and change
current operations, 233	narrative model, 232	analysis, 234

Review Questions

- Answer the following questions about current operations.
 - What should a team expect to learn from examining the current system?
 - What are the guidelines for deciding how much detail about current operations is needed?
 - It is very rare that no current system exists to be analyzed. Give an example.
- Answer the following questions about current situation models.
 - Why is it helpful to have both the narrative and the graphical models when they both reflect exactly the same system?
 - Who is the intended reader of the current system narrative?
 - Who is the reader of the current system graphical model?
- Consider the subject of information collection: teams usually interview their client(s) about current operations.
 - What other sources of information might be sought?
 - Under what circumstances might a client not want to share forms?
 - Why might a client not want you to talk to users?
- What level models are necessary in order to analyze the current operation?
 - How much detail should the DFD contain?
 - What is in an enterprise data model (EDM)? What does it leave out?

5. What information do you include in the graphical data model?
 - a. What metadata should be included with this model?
 - b. When should you include an ERD in the appendix and not in the body of the report or presentation?
 - c. What are the standard guidelines for the enterprise data model?
6. What information do you include in the graphical process model?
 - a. What metadata should be included with this model?
 - b. When should you include the DFD in the appendix and not in the body of the report or presentation?
 - c. What are the rules for DFD preparation?
7. Answer the following questions about retention and change analysis.
 - a. What is retention and change analysis?
 - b. According to the chapter, “The team reviews the description of the current operations, the problem analysis, and the proposed system features.” How does this work?
8. Answer the following questions concerning the project definition presentation.
 - a. What are possible objectives of the project definition presentation?
 - b. How does the team use the meeting to correct any errors or omissions in their understanding?
 - c. What are some techniques for generating discussion in the project definition presentation?
9. Upon completing the project definition stage,
 - a. What should the team do after the project definition presentation to complete that stage of work?
 - b. What are some of the milestones that should be included in the revised project plan?

Critical Thinking Exercises

Individual Exercises

1. Using the information on GB Video provided in Chapters 6 and 7, prepare a project definition report.
2. Prepare a project definition presentation for the report developed in exercise 1, above.

Group Exercises

1. Use your knowledge of your own college registration system,
 - a. Prepare a narrative of the current system.
 - b. Prepare a DFD and EDM that match the narrative.
 - c. Assuming that you have been asked to develop a computerized system to replace the advising system, prepare a retention/change analysis of the system.
2. Use the example of the motor vehicle problem 4 from the Group Exercises in Chapter 3, and the information you prepared in your answer to that exercise to prepare the following:
 - a. A project definition report.
 - b. A project definition presentation from the above report.

Proposed System

For many people, project definition tasks seem analytic, routine, and even mundane; finding a proposed solution feels creative and more satisfying. A typical team member views inventing the specifications for solutions as a natural task. Many teams begin to think about and identify satisfactory solutions from the first day of the project. Imagination, creativity, knowledge, and experience all play important roles in finding solutions. Team members know that their mission is to satisfy or give value to the client, and the client wants a solution for his or her problem. The proposed system activities of identifying and evaluating solutions focus directly on what the client wants.

Alas, the proposed system activities, regardless of the amount of creativity and enthusiasm the team invests, may offer little to the client unless the team builds carefully on the information from project definition. During project definition, the team identified the client's values, features, and constraints for the proposed system. The team expands and refines the features set forth by the client within the value and constraint framework to arrive at specifications for the proposed system and a recommended solution. Unless the team truly and fully understands what the client wants and values, the proposed solution may disappoint the client.

Chapter 8 explores the concepts and issues of developing specifications for a proposed system that will solve the client's problem. At this stage, the team determines conceptual specifications for the proposed system that the team can translate into different sourcing and infrastructure alternatives at a later stage. While most people can devise solutions using intuitive approaches, the team may benefit from applying more formal problem-solving methods. Helpful methods include heuristics, difference reduction, calculation, and applied experience. The team also may gain insight from such organizational frameworks as the value chain model.

As specifications emerge, the team needs to make another important decision: how best to document or record the specifications for use in future design or procurement activities. Good specifications provide a clear, unambiguous, and full description of the essentials of a proposed system that will allow the team to create and evaluate specific solution alternatives. The team may choose a combination of a narrative statement of specifications and such graphical data and

Chapter Eight

Proposed System Specifications

Chapter outline

Introduction

Goals and Outcomes

Concepts for the Proposed System

Problem-Solving Methods

Experience-based Methods

Trial and Error

Heuristics

Difference Reduction

Calculation and Optimization

Organizational Models

Design Approaches

Modification

New Design

Narrative Specifications

Narrative Format

Introduction

Data Specifications

Process Specifications

Organizational Specifications

GB Video Narrative Model

Graphical Process Specifications

Modified Data Flow Diagrams

The Context-Level DFD

The First Explosion MDFD

Additional Explosions

Graphical Data Specifications

Metadata Specifications

Object-Oriented Design Specifications

Use Case Diagrams

Class Diagrams

Summary

Key Terms

Review Questions

Critical Thinking Exercises

Individual Exercises

Group Exercises

References

INTRODUCTION

During project definition, the team performs mostly analysis activities. These activities include decomposing the situation into such pieces as strategy, problems, features, constraints, and operations; gathering information about each piece from the client; and analyzing the information. The team may bring the results from the project definition stage into the proposed system specification process including the:

- Statement of work.
- Project plan.
- Strategic alignment.
- Proposed system features and constraints.
- Problem analysis.
- Retention and change analysis.
- Sourcing options acceptable to the client.

Requirements specification for the proposed system marks the beginning of the synthesis phase of a project. **Synthesis** means combining separate pieces to form a coherent whole, that is, a proposed system solution that meets the client's requirements. During requirements specification, the team begins to assemble the pieces for the proposed system. Imagination, creativity, knowledge, and experience all play important roles during synthesis. Requirements specification builds directly on the existing prior work. The client identified the features and constraints that he/she feels are relevant to the proposed system. The team by means of diplomatic questions and suggestions tries to expand and refine the features mentioned by the client. If the team believes the client is overlooking or misconstruing an important feature, the team can ask the client questions for clarification.

This chapter assumes that the team will determine the **conceptual specifications** for the proposed system separately from sourcing and physical infrastructure alternatives, a useful practice when time and cost permit. The team applies problem-solving techniques and a variety of data, process, and object models to arrive at the conceptual specifications. Conceptual specifications represent solutions that the team can implement in different sourcing and infrastructure options. In the framework of this book, teams choose a sourcing option, such as to build or buy, and select a general physical infrastructure during the evaluation phase, which is covered in Chapter 9.

In practice, the nature and level of the proposed system specification process depends on the organization's policies, the complexity of the problem, and the team. For example, the team can tailor specifications to fit a sourcing option specified by the client or company policy: "Our company policy is to purchase packages for all non-mission-critical systems and build mission-critical systems in-house." When the team plans to build the new system, requirements specification may merge with detail design or coding and testing. When the team plans

to purchase the new system, requirements specification may merge with the identification and evaluation of packages.

Conceptual specifications and technologies may mix together or interact. The team may introduce or imply technology constraints without meaning to do so. For example, a data model for the proposed system that identifies foreign keys implies the use of a relational database. With prototyping or prototype-based design, the team uses a physical model of the system built with specific technologies to determine or refine the conceptual specifications. The conceptual specifications may dictate parts of the physical infrastructure. For example, a grocery sale system with a reasonable set of functional specifications probably requires bar-code scanner input to be cost-effective.

In some situations, organizational policy and/or the client may influence the physical infrastructure. For example, policy or the client may specify the use of the existing physical infrastructure or may specify the use of certain products, for example, UNIX operating systems or Intel chip servers. Policy may require the use of such higher-level languages and development systems as Microsoft Visual Basic or Oracle development tools. These tools generate and/or work best in specific logical and/or physical structures; for example, Visual Basic works with an event-driven rather than a batch-sequential logical structure, and Oracle tools assume an Oracle relational database.

Creating conceptual specifications gives the team and client significant flexibility. Conceptual specifications translate into a variety of logical and physical environments. The sourcing option selected might include to build in-house, to buy a packaged application, or to outsource to an application service provider. Technology environments might involve mainframe or netcentric architectures, relational, network, or other data structures, and a broad set of available servers, operating systems, and data storage devices. If the team can demonstrate that sourcing options or infrastructures other than the specified ones lead to better cost-effectiveness, clients may change their minds and policies can have exceptions.

GOALS AND OUTCOMES

The goal of the proposed system phase is to assure that the conceptual specifications for the proposed system are complete and correct before the team expends any significant effort on the logical and physical design—a philosophy of “make it right; then make it work.” Developing a complete and correct understanding of the conceptual specifications of the system can consume substantial time and effort and may involve additional information gathering activities. For example, the team may conduct additional interviews with clients and users and/or observe the users at work. The team may conduct prototyping sessions at which clients and/or users try out a prototype or package. By whatever appropriate processes, the team must identify the desired features and outcomes and translate them into conceptual specifications for the proposed system.

During this phase, the team answers questions that include (1) what functions or processes should the system include to provide the features that the client

wants and (2) what data are required to support or operate the included functions? The team may answer these questions by preparing such deliverables containing specifications for the proposed system as:

- A narrative statement.
- Graphical process and data models.
- Graphical object models.

The team creates narrative and graphical models of the proposed system specifications because models offer an effective way to create and communicate the requirements. Models provide a standard format that may facilitate communication between team members and sometimes with clients. The amount and level of modeling tends to be both more detailed and more extensive than for the current situation because the proposed system models serve as input for the system delivery phase. The conceptual specifications models provide the base for the logical and physical specifications for the programs and data. However, the logical and physical designs generally involve an even greater level of detail than conceptual specification.

The narrative model describes the proposed system specifications in natural language, but follows a specific format to encourage completeness and to facilitate communication between team members. A **conceptual data model (CDM)** can illustrate graphically the data environment for the proposed system including all of the entities, relations, and attributes. A CDM includes all of the entities that will define data stores in the new system, that is, it includes the entities for data stores inside the new system plus the entities for external data stores that the new system uses to retrieve or store data.

The team may use one of a number of process models to specify the program structure and logic. Many projects will create the conceptual process model with **modified data flow diagrams (MDFDs)**. MDFDs work well as conceptual process models for a wide range of project types, but other process models may work better for some projects. When the client wants a Web site or other dialog-driven system, a page navigation map may offer the most effective process model (see Chapter 11). Alternatively, the team may choose to create **object-oriented design (OOD)** diagrams to represent the conceptual specifications. The team should discuss their plan for requirements specification and the proposed data and process models with the team manager before investing time and effort.

CONCEPTS FOR THE PROPOSED SYSTEM

As previously noted, the team must solve a synthesis problem to determine specifications. To synthesize a solution, team members must combine the appropriate features into the conceptual **specifications for the proposed system**. Most people find inventing the specifications for solutions a natural task. Many teams start to think about and identify satisfactory solutions from the first day of the project. However, informally generated solutions and specifications come with some possible hazards. The informal solutions may not deal with all of the client's requirements and may involve higher costs and lower benefits than other

possible solutions. The team is well advised to follow or at least consider more formal approaches to generating specifications. The more formal approaches include problem-solving methods, organizational models, and design approach models.

Problem-Solving Methods

Much thought has gone into and much has been written about **problem solving**. Team members solve hundreds of problems every day without even thinking about them, such as what to wear, where to eat lunch, what to do first when arriving at work, and so on. As problems become more complex, the additional structure provided by the use of problem-solving methods may help. The problem-solving methods discussed in the following materials apply to all of the aspects of the synthesis phase as described in Chapters 8 through 13.

To the casual observer, teams appear to come up with design decisions by some unknown gestalt. Actually, the team, perhaps without conscious intent, applies a mix of problem-solving methods to make the design decisions. The team can choose from a large array of problem-solving methods, but most methods fall into one of several classes:

- **Experience-based methods** that use prior experience or experiences of other organizations to derive solutions.
- **Trial and error methods** that try out different solutions.
- **Heuristic methods** that follow a set of rules that “seem to work well” but do not guarantee a best solution to a problem.
- **Difference reduction methods** that identify and try to reduce the differences between the current state and the desired state.
- **Calculation and optimization methods** that use mathematical procedures to find a solution.

The analyst may use a combination of several methods to solve the system synthesis problems.

Experience-based Methods

Experience-based methods find frequent use in information systems work. The analyst may bring direct experience from solving similar problems or may use ideas learned from colleagues. Sometimes an analyst will look at a solution in another organization. The literature, including books, trade journals, and professional journals about information system and organization solutions, often provide a rich source for experience-based solutions. For example, most people have rented a videotape or DVD and know from their own experience and/or from observation and literature that most video rental systems contain member, rental, and return modules; use bar-code readers to enter the video ID number, and so on. Most teams facing the GB Video problem probably would use experience to arrive at an initial set of specifications for a proposed system. The team also might visit a video rental company to learn about their system.

Brainstorming offers another experience-based variation for problem solving that can generate innovative solutions. During a brainstorming session, a group

of people including team members, clients, and perhaps others, use experience to come up with a broad set of possible solutions. No attempt is made to evaluate the solutions; the goal is to generate a wide-ranging set of ideas. Brainstorming can take place in a standard face-to-face meeting or with the use of group support system (GSS) technologies. Sometimes brainstorming leads to innovative solutions that differ from the solutions used by most organizations.

Trial and Error

Trial and error methods relate closely to experience-based methods. With trial and error, the analyst may use experience to determine an initial “trial” solution. If the initial solution is satisfactory, the problem is solved. If the trial solution is less than satisfactory (an error), then the analyst looks for another solution to try out. Ideas undergo a trial in a variety of ways. The analyst may just think about a solution to determine if it is satisfactory or may seek the opinion of other team members and/or the client. Prototyping provides a more formal structure in which to evaluate solutions in a trial and error approach.

Heuristics

In information system design, heuristic problem solving normally consists of finding and applying “best practices.” **Best practices** represent generalizations of experience, that is, ideas that appear to provide benefits learned from experience at a number of organizations. For example, using an SDLC approach to plan a system project is a heuristic. Often heuristics build on pieces of theory or principle, but the theory framework is incomplete and does not guarantee a best solution. Some typical heuristics or best practices for solving conceptual specifications problems in information systems include:

- *Design the system to capture input data only once.* After the initial capture, retrieve the data from storage when they are needed. (Underlying principles: data input is expensive and error prone; single input should reduce cost and errors.)
- *Design the system with single point data storage for possible use by multiple application programs.* All of the customer data is stored in one and only one data store. (Underlying principles: duplication of data or multiple data stores for the same data can lead to data integrity problems and can increase the costs for updating and maintenance.)
- *The system automatically should generate and insert standard input data when possible.* For example, the system can generate current date and time, transaction number, or customer number for a new customer. (Underlying principle: the system can generate these data at lower cost and higher consistency than manual input.)
- *The system should perform all calculations.* (Underlying principle: the system can perform calculations at lower cost and higher accuracy than people.)
- *The system should verify data about third parties.* Third parties include customers, vendors, employees, and others; data should be verified on every contact with a third party. (Underlying principle: people tend to omit or delay reporting changes in their data.)

- *Error checking of input data (online, real time if possible) represents a good investment in a system.* (Underlying principle: The cost to prevent erroneous data from entering the system is less than the cost to correct bad data and/or repair the damage that bad data can cause.)

Hundreds or thousands of other best practices exist. While best practice methods provide helpful guidelines for requirements specification or other parts of synthesis, a “best practice” may or may not constitute the best solution. Conditions may exist that cause a best practice to provide a less than satisfactory solution. For example, extensive error checking may increase costs, slow down input, antagonize the third parties, and add little if anything to data integrity.

Difference Reduction

Difference reduction methods find widespread use in information’s system design. Many times during design, the analyst, either directly or indirectly, compares the current state to the desired state and looks for ways to move the current state closer to the desired state. The following list (adapted from Zuboff, 1988) provides one characterization of general operations to move from the current state to a desired state for an information system.

- **Automate.** Replace manual activities in the current situation with functions performed by a computer-based solution and/or replace current information technologies with more cost-effective ones—a physical design issue.
- **Inform.** Provide new information that is not available in the current situation to such users as managers, employees, vendors, and customers.
- **Transform.** Change the functions, structure, and/or performance for organizational processes in the current situation.
- **Discover.** Allow users to learn things about the organization and/or its environment for which the current situation does not allow discovery.

Many times, the team will use a combination of the operations to move from the current situation to the proposed system. For example, “Automate” is an important operation to move from the current situation at GB Video to the desired solution. The GB proposed system also “Transforms” the way the organization does business by allowing access to rentals only for confirmed members, and “Informs” by providing new information on members to marketing. The new system may allow Purchasing to “Discover” customer rental patterns.

An obvious application for difference reduction involves looking at the problems in the current operation and finding ways to move to a desired “problem-free” state. The difference reduction table shown in Table 8.1 lists problems identified in Chapter 7 for the GB Video example and the possible actions or operators that the analyst can apply to move toward the desired state.

The analyst also can apply difference reduction to the features that the client wants in the proposed system compared to the current operation. The difference reduction table shown in Table 8.2 uses the proposed system features identified in Chapter 6. A number of the differences relate to logical and physical design; Chapters 9 through 13 deal with some of these design issues.

TABLE 8.1
Difference
Reduction
Table Based
on Problems

Problems in the Current Operation	Reduction Action or Operation to Arrive at a Problem-free Proposed System (PS)
Manual system leads to high cost	Physical design issue—automate
Long lines, slow checkout	Physical design issue—automate
Accessing manual files causes delays	Physical design issue—automate
Incorrect rental charges	Include a better calculation function in the PS
False or missing member numbers	Access the rental function only from the member function in the PS
Nontimely overdue notices	Include a better overdue notice function in the PS

This comparison tends to duplicate part or much of the problem table but may contain some new information. A team may wish to combine the problem and feature tables.

Calculation and Optimization

Standard mathematical approaches apply to solving some problems, especially physical infrastructure problems. For example, a GB Video customer record might consist of 250 bytes. Assuming that the system after blocking and other storage considerations can utilize 80 percent of the storage, how much storage is required for 1,000 customers? This problem becomes a simple calculation using the following formula:

$$\text{Total data storage} = \frac{(\text{Customer record size} \times \text{Number of customers})}{\text{Utilization}}$$

$$\text{Total data storage} = \frac{(250 \times 1,000)}{0.8} = 312.5 \text{ KB}$$

TABLE 8.2
Difference
Reduction
Table Based
on Features

Current Operation	Client-Desired Proposed System Features	Reduction Action
Member, rental, and return functions	Faster, simpler, member rental and return functions	Redesign rental, return, and member functions
All manual data entry	Automate data entry where possible	Physical design issue
A lot of clerk time	Less clerk time	Physical design issue
Duplicate data entry and data storage	Reduce the cost and errors for data entry	Include single point entry and storage
Manually send data to accounting	Eliminate manual data transfer	Give accounting direct access to data stores
Credit card and rental form processing are entirely separate	Save the cost of separate processing of the rental form and the credit card	Integrate credit card processing into the rental function
Rental and Return system contains two reporting functions	No reporting functions in the rental and return system	Transfer reporting functions to accounting
Manual overdue notices	Automated overdue notices	Physical design issue

Optimization methods consist of mathematical processes for finding the conditions that result in the minimum or maximum value of a function. Common optimization methods include differential calculus, difference techniques, mathematical programming, and enumeration. While optimization techniques are difficult to apply to many problems, they can provide powerful solution tools for some problems.

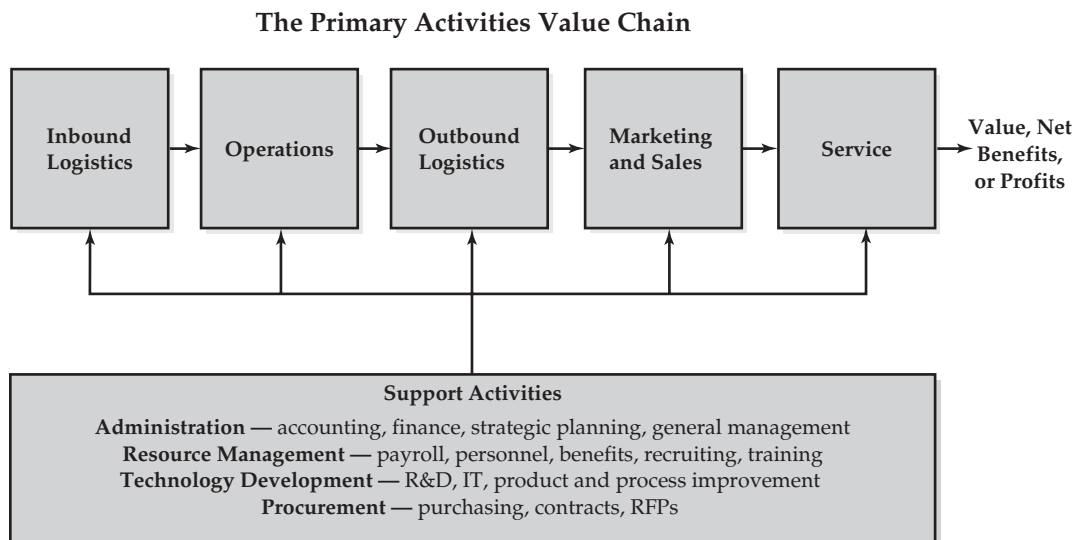
Organizational Models

The **Value Chain Model** of an organization can provide guidance on developing specifications for the proposed system. Michael Porter developed a value chain model in his analysis of effective business strategies (Porter, 1985). The value chain follows the system decomposition model used by such systems analysis tools as DFDs: break the value-producing activities of the organization into components that allow managers to identify and focus on solvable problems. A value chain represents the value production function for the organization—the activities that result in the total value, net benefits, or profits produced by the organization.

The value chain model shown in Figure 8.1 identifies five primary activities and four support activities. The five primary value chain activities in the upper half of the diagram consist of:

1. *Inbound logistics*. The acquisition, transportation, and storage of resources needed to produce the desired good or service.
2. *Operations*. The activities that actually create the product or service.
3. *Outbound logistics*. The actions necessary to deliver the product or service to a destination, such as warehousing, transportation, and so on.

FIGURE 8.1 The Value Chain Model



4. *Marketing and sales.* Advertising, promotion, and other activities that make potential customers aware of the results and more likely to use them.
5. *Service.* Activities that support the product or service after initial delivery to a customer, such as training, spare parts, maintenance, and updates.

The support activities appear in the lower half of the diagram. The arrows show the flows of value.

Most of the time, team members can frame their system problem into a form that addresses value through the primary value chain. Problems in any of the primary activities can reduce the total value produced by the organization. The objective of value chain analysis is (1) to identify aspects in the value chain for the current situation that constrain the value production function; and (2) to identify proposed system features that will eliminate or reduce the constraints.

To use value chain analysis, the team carries out the following steps:

1. Define the product or service related to the goals and objectives identified during strategic alignment. If the goal is to increase profits at GB Video, the related product is video rentals.
2. For the product or service related to the goal, identify the key subactivities for each part of the primary value chain. Different people may put subactivities in different places. The important part is to identify the subactivities. The value chain serves only as a guide for thinking about the activities. All or most of the relevant subactivities may fit in one area of the value chain. In the GB Video example, many of the subactivities fit into the operations activity of the value chain.
3. Classify each subactivity as (1) a strength for which there is little reason to invest further effort; or (2) a weakness for which additional effort may have a large payoff. The idea is to find the most promising areas for improvement.
4. For the most promising areas, create features for the solution that will eliminate or mitigate the weaknesses and thereby add value to the chain.

Although features of the proposed system may contribute in some way to all of the primary activities, the team should focus on the most important contributions. These key contributions drive the major decisions and also are most likely to attract the support of the sponsor. Teams often are tempted to focus on such technology features as, “The Web interface will allow people to order their own videos online and reduce the cost of outbound operations.” At the conceptual stage, the team should try to focus instead on the features that provide the desired functionality and performance for the new system.

Design Approaches

Design approaches determine the route or processes used by the team to move from the current situation to the proposed system. Clients often request one or both of two broad design approaches for the new system: modification and/or a new design.

Modification

If the client requests a **modification of the current system** or a new system that closely resembles the current system, the team can modify and extend the current operation materials to incorporate the features of the new system. The disadvantage of this approach is that the team may miss or ignore changes that could improve greatly the functionality of the system.

The GB Video example represents primarily a modification of the current manual system. Both the current operation and the proposed system use the same basic functions and data. The biggest changes occur with respect to the logical and physical design; these changes include automating the manual system, changing the data structure, and rearranging responsibilities. At the conceptual level, the MDFDs for the proposed system should look similar to (but not identical to) the DFDs for the current operation. In short, a more detailed analysis of the current operation can provide such significant information for the proposed system as data formats for attributes of all of the entities and detailed procedures for rentals, returns, and members.

New Design

If the client requests a new system that differs significantly in function from the current situation, the team may wish to construct the new narrative from the strategic goals and objectives and from the desired features and operation of the new system. With a new design, the team avoids the design bias that may result from starting with the current operation narrative and graphical models. After the conceptual design for the proposed system is complete, the team may review the models of the current operation to make certain that no key features were omitted. If no current system exists, new design is the only option.

For the GB Video example, a team probably would not use a new design approach because much of the data and processes remains the same. In the new system, customers will request and return videos with the same processes and flows as before. New design aspects enter into the GB Video example mainly at the logical and physical level. To meet the client goals, the logical data structure probably will change to a relational model, and physically, the system will change from manual operations to bar-code scanning, keyboard input, and computer processing. Analysts tend to think about all of these issues at the same time. As a result, analysts working on actual problems in organizations often combine the conceptual, logical, and physical design phases for small projects.

If the nature of the GB Video system changed substantially from the proposed system, then new design becomes more relevant. For example, the client at GB Video might ask the team to create a Web-based video rental service. The client wants to use the Web site to rent videos. The customers will select videos on the Web and then pick up the items at their convenience or receive them from a delivery service. The new operation will differ substantially from the current operation. The current operation still can provide information on the attributes of the entities for the proposed system, but clearly the proposed system will

contain some new processes and functionality even at the conceptual level. This scenario also provides an example of technology-driven design. The introduction of Web technology results in changes in functionality and processes. Chapter 11 contains illustrations that show how the resulting system might work.

NARRATIVE SPECIFICATIONS

The narrative model provides the specifications for the proposed system in a text format. The **narrative specifications** paint a comprehensive and precise natural-language picture of both the rationale and the features for the proposed system. The proposed system narrative specifications may differ significantly both in structure and content from the narrative for the current operation. The proposed system narrative applies the results of problem-solving techniques, organizational models, and design approaches to synthesize the conceptual specifications. In the proposed system narrative, the events, data flows, processes, and data specifications reflect those needed to meet the client's goals, instead of the actuality of the current operation. In addition, the team provides the specifications in sufficient detail to construct the program and data schema during detail design in-house or to contract with an outside vendor for the purchase option.

Narrative Format

For completeness and ease of comprehension, the proposed system narrative specifications follow a standard format or a structure. The analyst may wish to include the following items about the proposed system in the narrative in language that the client and an outside vendor, if applicable, will understand:

1. An introduction.
2. A discussion of the problem-solving methods and rationale that the team applied to determine the specifications. Normally the team discusses the problem-solving rationale as part of the data and process discussion.
3. The proposed system process and data specifications or object specifications.
4. Any organizational changes required to use the proposed system.

Although the team should try to avoid any commitment to specific technologies in the conceptual specifications, field projects focus on exploring the design and use of computer-based information systems. During the evaluation phase a team might conclude that a manual system appears most cost-effective. However, the great majority of projects will result in a computer-based system and the specifications often assume explicitly or implicitly that the solution will involve a computer-based system.

The order in which the team prepares the data and process specifications depends on team preferences and project content. With systems that focus mainly on data, for example, building a database that will support a number of applications or a data warehouse, the team may wish to start with specifying the data structure. If the project involves complicated processes, such as a reservation system, the team may choose to start by specifying the process structure. In any

event, preparing the narrative, data, and process models often involves an iterative process. The team starts by specifying the data structure and probably will discover additional data specifications during process specification. When the team prepares the **graphical model specifications**, the team probably will discover changes for the narrative or vice versa if the team starts with graphical models.

Introduction

The introduction to the proposed system specifications restates briefly the problem and the major issues or goals for the team. The introduction may include the content of the project statement to clarify again what the team is trying to do. The introduction also identifies the content of the rest of the specifications materials.

Data Specifications

At this stage, the team generates the conceptual data specifications for the proposed system. The specifications may closely resemble the current operation or may differ significantly. The team should explain the rationale for retaining and/or changing the data specifications using problem-solving methodology. The team identifies only the content of the data specifications at this stage; decisions on the logical and physical structure come in the next step. Normally the narrative specifies the entities for the new system, most or all of the attributes, cardinality, and primary keys or identifiers. The analyst should include information about file sizes or expected number of instances if available. If the proposed system materials contain graphical data models, the narrative should contain explicit references to each of the graphical models.

Preparing the data specifications for a proposed system involves synthesis or creativity. As discussed, experience, difference reduction and best practices, may give the analyst ideas for the proposed system. Some thoughts on applying these methods specifically to determine data specifications include:

- Look at the data specifications for the current operation. Keep the entities and attributes that appear useful, if any.
- Review the problems identified by the client and look for the new entities and attributes that will reduce or eliminate the problems.
- Determine the entities and attributes required to support the client-specified goals and functionality for the proposed system. If the client wants to send mailings to customers, the system must contain the customer mailing addresses as part of the data specifications. If the client wants to know how many customers are over 65, then the data need to contain birth date.
- Look at the data specifications used by competitors or other similar organizations for this kind of system. Looking at other organizations Web sites may reveal a lot about the data specifications in use.

Team members always have a professional obligation to the client, namely the obligation to use their information system and organizational expertise to help the client. Clients often experience difficulty specifying the data required for

operations, decisions, and analysis. The team should do more than ask what the client wants. For example, the team can create options and suggestions and then ask the client to comment on them. If the team builds a prototype, the team probably will discover that some data needed to make the prototype operate are missing from the original specifications.

Process Specifications

The team specifies the processes in the proposed system required to meet the client's goals. The process specifications tell the story of how the new system will work, including who does what and when. Normally, the paragraphs in the process section correspond to modules in the graphical process model or operations in the object model. When the proposed system materials contain graphical process or object models and/or input and output specifications, the text should contain references to the graphical models and specifications.

The team also applies problem-solving methods to determine the process specifications. Suggestions for applying problem-solving models that are specific to process resemble those for data and might include:

- Looking at the processes in the current operation and keeping the useful ones, if any, as part of the proposed system specifications.
- Reviewing the problems, features, and goals identified by the client and identifying new or changed processes that will reduce problems and/or meet goals.
- Identifying all of the outputs to externals, such as shipping notices, purchase orders, and manufacturing schedules. Each output activity to any external requires specification of one or many processes.
- Examining the ways that data enter the proposed system. An input process is required any time that data enter the system from an external.
- Specifying the processes that are required for all the data store operations when the system stores, retrieves, updates, or deletes data.
- Checking carefully that processes exist in the specifications to move data from the point of origin all the way to the point of use. Any missing process link in the chain will cause an integrity error.

The specifications may contain both **mandatory processes**—those processes that must be present to provide the minimum functionality required by the client and desirable or **optional processes**. One common set of processes, error checks, often involve both mandatory and optional processes. A check to assure that the credit card number is present in the data record for a new GB customer probably is mandatory. An error check to determine if the zip code matches the customer's city and street address may be optional. Other kinds of desirable processes include processes that simplify the task for the user, for example, the automatic transfer of member data from the member process to the rental process, or processes that enhance system usefulness for a customer. During the evaluation stage (presented in Chapter 9), the team and the client will explore which optional processes to include in the final system specifications.

Organizational Specifications

The organization specifications identify the relevant organization issues for the proposed system. At a minimum, the organizational specifications define the sources of external inputs and the recipients of external outputs. Such graphical representations of the proposed system specifications as DFDs and use case diagrams show this information. Another common organizational issue is to determine who has the authority and responsibility to specify data and process and to request or approve changes. The organizational specifications may include identification of the organization unit(s) or title(s) that may access and change data, perform a process, and/or is responsible for its correct operation. If the proposed system materials contain graphical organizational models or charts, the text should contain references to the models.

GB Video Narrative Model

A sample narrative model for the GB Video proposed system specifications appears in Figure 8.2. The narrative model in Figure 8.2 provides only an illustration of specifications for a proposed system. While the sections may remain the same for many field projects, the content and emphasis may vary widely. Each team should tailor their narrative model specifications to the specific problem.

GRAPHICAL PROCESS SPECIFICATIONS

The team may choose to represent specifications graphically with a variety of graphical process models. The graphical process model must:

- Represent correctly the specifications for the process.
- Help to define the structure of the final program or code.
- Match the specifications in the narrative model.

Modified data flow diagrams may represent a good model especially for event-driven, transaction processing or batch processing situations. As noted before, the team also may use an object model to define the structure of both data and process. In some situations, the computer program itself or a prototype of it serves well as a process model.

Modified Data Flow Diagrams

Modified data flow diagrams (MDFDs) include all the features and rules of DFDs plus some additional rules that help the analyst to focus more clearly on specifications for the proposed system. The additional rules link the ERD for the proposed system to the proposed system DFDs, introduce the concept of process triggers, and add time as a dimension to the DFDs.

A **trigger** for a process is the event or thing that causes the process to carry out the code, logic, or actions it contains. Every process must have a trigger that causes the process to begin to operate. Possible triggers include:

- *A data flow into the system from an external*, for example, the flows associated with a customer providing member data or with a customer returning

FIGURE 8.2 Narrative Model for GB Video

Introduction

The proposed computer-based Rental and Return System will focus on improved customer service and lower handling costs for each transaction. This narrative addresses only the conceptual specifications for the proposed system. The specifications described in this model derive from the problems, features, and functions identified by GB Video during project definition. The team applied experience, difference reduction techniques, and best practices to arrive at the conceptual data and process specifications described in both narrative and graphical form in the following materials. The team will recommend sourcing and physical features in subsequent materials.

Problem Solving

The proposed system builds on a modification of the current operation. The team members used their experiences with video rental systems plus information in the literature to arrive at a tentative set of features for the proposed system. The team incorporated the following best practices in the desired system:

- Capture rental and return input data only once. After the initial capture, retrieve the data from storage when they are needed.
- Single point data storage of rental and customer data for use by the rental and return system and by Accounting.
- Automatic generation and insertion of rental number and customer number for a new member.
- System calculation of tax and total cost.
- System verification of customer data on each contact with a customer.

The team also used difference reduction. The difference reduction tables for problems and client-desired features appear below. All of the features suggested by these tables are incorporated in the proposed system narrative and graphical models.

Problems in the Current Operation	Reduction Action or Operation to Arrive at a Problem-Free Proposed System (PS)
Incorrect rental charges	Include a calculation function in the PS
False or missing member numbers	Access the rental function only from the member function in the PS
Nontimely overdue notices	Include an overdue notice function in the PS

Current Operation	Client-desired Proposed System (PS) Features	Reduction Action
Member, rental and return functions	Faster, simpler, member, rental and return functions	Redesign rental, return, and member functions
Duplicate data entry and data storage	Reduce the cost and errors of data entry	Include single point entry and storage
Send data to Accounting	Eliminate data transfer problems	Give Accounting access to data stores
Credit card and rental form processing are entirely separate	Save the cost of separate processing of the rental form and the credit card	Integrate credit card processing into the rental function
Rental and return system contains two reporting functions	No reporting functions in the rental and return system	Transfer reporting functions to Accounting

Process Specifications

The proposed GB Video Rental and Return System contains the following functions: (1) Member—create a new member record or update an existing one; (2) Rental—rent videos; (3) Return—return videos; and (4) Overdue—create overdue notices for videos that are overdue. These functions resemble the ones in the current operation. However, using the problem-solving methods described above, the team introduced a number of changes to achieve the customer service and cost goals specified by the client. The client considers all of the functions described in the narrative mandatory.

The customers of GB Video identify the videos that they wish to rent and go to a checkout position. To improve the likelihood that GB rents only to customers who are members, the member option must be selected at the beginning of every rental transaction.

Member

The member process is triggered by a customer request to (1) rent videos and/or (2) become a member. If the customer has and knows the customer number, the number is entered; the system retrieves the record from the Customer data store and displays the customer data. If the customer does not have the number, the customer can provide a telephone number (or a name and zip code). The system tries to retrieve the customer's record from the Customer data store.

If the system is unable to retrieve the customer record or if the customer is not a member, the new member subprocess is initiated. The system will create a new member provided the person has a credit card, telephone, and government-issued ID. The customer supplies the customer data and the data are entered. The system generates a customer number, creates a membership card output, and gives the output to the new member. The system creates a record in the Customer data store for the new member.

Once the appropriate customer record is available, the customer is shown and asked (1) to verify the name, address, telephone, and credit card data and (2) to report any changes or corrections. This subprocess increases the likelihood that the system will contain current information for the customer. Any change data are entered and the customer data store is updated. When a verified customer record is available and the customer wishes to rent, the member process triggers the rental option and the member data are sent to the rental process. The rental process can be accessed only from the member process; the rental process cannot be accessed directly.

Rental

The rental process is triggered by and only by the member process. The rental process accepts the member data from the member process and generates a new rental transaction number and the rental date. The customer provides the video number and the proposed return date, or the due date. The video number and the due date are entered into the system. The system retrieves the video data from the Video data store and based on the due date calculates the rental price for each tape or DVD rental. This process is repeated for each video. After all the videos are processed, the system adds the rental price for each video to get a rental subtotal. The system multiplies the rental subtotal by the tax rate (state sales tax + county sales tax + city sales tax) to get the tax. The total rental cost is the rental subtotal plus the tax.

The payment type—cash, check, or credit card—is supplied by the customer and entered into the system. If payment is by credit card, the credit card data are entered. The system sends the credit card data and the total rental cost to the credit card processor. If the transaction is approved (or is for cash or a check), the system “commits” the rental and line records; the records are created in the data stores. The rental number, date, clerk number, pay type, and credit card data go to the rental record. The due dates go to the line records for each video.

The system then generates a receipt for the customer. The receipt data includes all of the data supplied by the customer, sent from the member process, and generated by the system during the rental transaction. The system also retrieves the title data from the Title data store and includes the name or title in the receipt data.

Return

When a video is returned, the video number is entered into the system. The system retrieves the corresponding line and rental records, enters the return date, and calculates overdue charges if any. The customer may charge the overdue fee to the credit card used for the rental or may pay by cash or check. About 95 percent of the time, customers return videos with no payment type input, for example, drop them in a return box. In the absence of customer input on payment, the system retrieves a credit card number from the customer record and processes the overdue charge against the credit card. The system records the charge and payment type in the Line data store. The system may create a return receipt for the customer.

Overdue

After the overnight returns are processed, a clerk instructs the system to run the overdue program, that is, it triggers the overdue function. The system processes the Line records. For each video that is two or more days overdue (i.e., today's date – due date ≥ 2), the system retrieves the rental record for the video, retrieves the customer

record for the rental, generates an overdue notice, and sends the notice to the customer. When a video is 14 days overdue, the system retrieves the customer's credit card number from the Customer data store and the video cost from the Video and Title data stores, charges the customer's credit card for the amount of the cost of the video, and sends a notice informing the customer of the charge. If a customer has a complaint about overdue charges, the complaint is handled and processed by Accounting.

These processes are represented graphically in the data flow diagrams in the next section.

Data Specifications

The new computerized system will contain data about Customer, Rental, Line, Video, and Title. The Customer entity will contain customer number, name, address, telephone, credit card number, and expiration date. If a member does not rent a tape for 24 months, the record is deleted and the person must rejoin as a new member to rent tapes. GB expects to start with about 4,000 members and this number will grow over time.

The Rental data will contain rental number, rental date, clerk number, payment type (cash, credit card or check), and for credit card payments credit card number, expiration date, and the credit card approval code received from the credit card company. About 80 percent of rental payments consist of cash, 10 percent checks and 10 percent credit cards. The Line data for each video rented will contain line number, due date, return date, overdue charge if any, and payment type for the overdue charge. An average rental consists of two videos, requiring two lines, but some rentals may consist of 10 or more videos.

At the end of each week, the Accounting system will process the Rental and Line data, will transfer the data on completed rentals to a data warehouse, and will delete the records for completed rentals. On average, the rental file is expected to contain about 800 rental records just before processing by Accounting.

The Video data contain the rental fees for each video number. A video may have three fees—the one-day rental fee, a lower fee for additional days, and a special fee for a weekend and/or holidays. Since GB closes on Sundays and holidays, the special fee encourages members to rent several videos for Sunday and holiday periods. The Title data contain the title number, name of the video, the vendor code, and the cost paid by GB to acquire the video. Title and Video define data stores that are accessed by the Rental and Return system but are created and maintained outside of the system.

This data structure is represented graphically in the conceptual data model in the next section.

Organizational Specifications

The organizational sources and destinations for external data are shown in the context-level DFD in the next section. The sales clerks, who report to each store manager, have the authority to retrieve, create, or update records in the Member, Rental, and Line data stores and to retrieve records in the Video and Title stores. The Accounting department at headquarters has the authority to retrieve records from all stores and to delete records in the Customer, Rental, and Line stores. Accounting produces all reports on rental activities. Records in the Title and Video data stores are created, updated, and deleted by Purchasing at headquarters.

a video. Any external request that causes a process to begin operation is a trigger.

- *A data flow from another process.* A flow from another process always triggers the receiving process. A process-to-process flow can occur only with a process or subroutine call.
- *A control flow as a result of time, a count, or other system condition.* A control message to prepare a report at the end of the month triggers the monthly report process.

Once the concept of triggers is added, time becomes a dimension of DFDs; in other words, a process is executed when and only when the trigger event occurs. One process can pass data to another process only when the second process executes immediately after receiving the data. If a time interval occurs between the two processes, the first process must place the data in a data store and the second process retrieves the data from the store at a later time. Sometimes, the first process sends data to an external, and the external at some later time may provide the data as input to trigger a second process.

Additional rules for MDFDs include the following:

- **DFD/ERD integration.** *All of the entities and only the entities on the conceptual ERD for the system appear as data stores in MDFDs. The contents of a data store in an MDFD is defined by the attributes of the entity.* This rule binds tightly together the conceptual ERD and the DFDs for the system and clarifies the specification process. During logical and physical design, an analyst or database administrator may decide to create data stores (tables, views, or files, etc.) that include attributes of several entities.
- **Process triggers.** *Every process on the MDFDs must be triggered by (1) a data flow from an external; (2) a data flow from another process; or (3) a control flow.* The trigger rules help the analyst to understand how the proposed system works and lay the foundation for the eventual physical design.
- **Time and immediacy.** *When a data flow goes from process A to process B, the flow always is the trigger for process B, that is, process B starts immediately when the flow arrives. A and B are sequential processes and B always takes place immediately after A.*
- **Labeling.** *External data flows that trigger a process are labeled by placing an asterisk (*) at the beginning of the flow name or by another symbol as noted on the diagrams.* Only some of the data flows from externals serve as triggers and thus trigger flows need a symbol. Process-to-process flows do not need a special symbol because they always serve as triggers.

The guidelines for creating MDFDs include these:

1. Observe the standard rules, features, and symbols for DFDs and the rules for MDFDs. Chapter 5 contains a full set of rules for DFDs.
2. Some analysts may wish to create the ERD before drawing the MDFDs; others may wish to use the MDFDs as a guide to selecting entities for the ERD.

3. Use the narrative to identify the data stores that will correspond to the entities on the ERD for the proposed system; the processes or functions; and the externals and data flows. The narrative is used as the source of the metadata for each process module.
4. Generate a first explosion MDFD and additional explosions in adequate detail to map to program modules and to match the narrative. Create a context-level diagram if desired.
5. Review the data stores to determine if they are inside or external to the system boundary. Stores internal to the system must conform to the data completeness rule, that is, each store must connect to at least one input and one output data flow. Some of the data stores may lie outside the boundary of the new system; as such, they are external data stores controlled and/or used by another system. External stores use the external symbol to connect to an input or output flow.
6. Identify a trigger for every process—the event that causes the process to begin to execute. When an external data flow corresponds to a trigger, place an * in front of the flow name as provided for in the Label rule. Data flows to and from stores and to externals may not serve as triggers. For processes that are triggered by a system condition, the trigger action can be shown using a dotted line for a control flow from an external called System.

The Context-Level DFD

The GB Video proposed system context-level DFD is shown in Figure 8.3. The process box contains the name of the system. The externals and data flows to and from externals correspond directly to the information presented in the narrative.

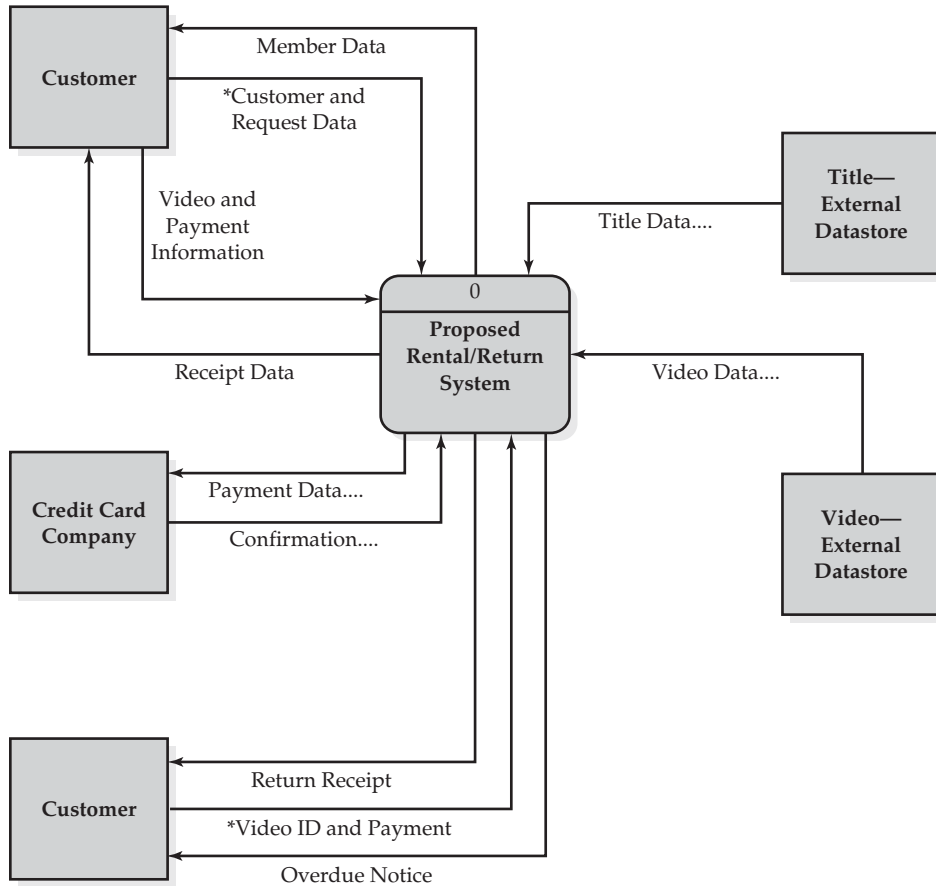
In Figure 8.3, data flows that go to more than one process on the first explosion are shown with several dots following the flow name. This convention eliminates the need to show the multiple flows on the context level and adds to clarity by simplifying the diagram. Using this convention, the context DFD and the first-level explosion DFD are consistent or balanced; in other words, all the flows and only the flows to and from externals on the context diagram appear on the first explosion. Two data stores, Video and Title, appear on the context level as externals because they are outside the rental/return system. The Video and Title data stores are created, updated, and managed by another organization.

The two external data flows that trigger processes are marked with an * at the beginning of the flow name: *Customer and Request Data and *Video ID and Payment. As noted in the rules, the flows from the system to an external never serve as triggers. The four other input flows from externals occur in response to actions from an ongoing process and at the conceptual level do not trigger the process. At the physical design level, these flows may trigger subprocesses.

The First Explosion MDFD

The analyst begins by selecting the major processes to show in the first explosion. Most field projects contain two to six or so major processes; large

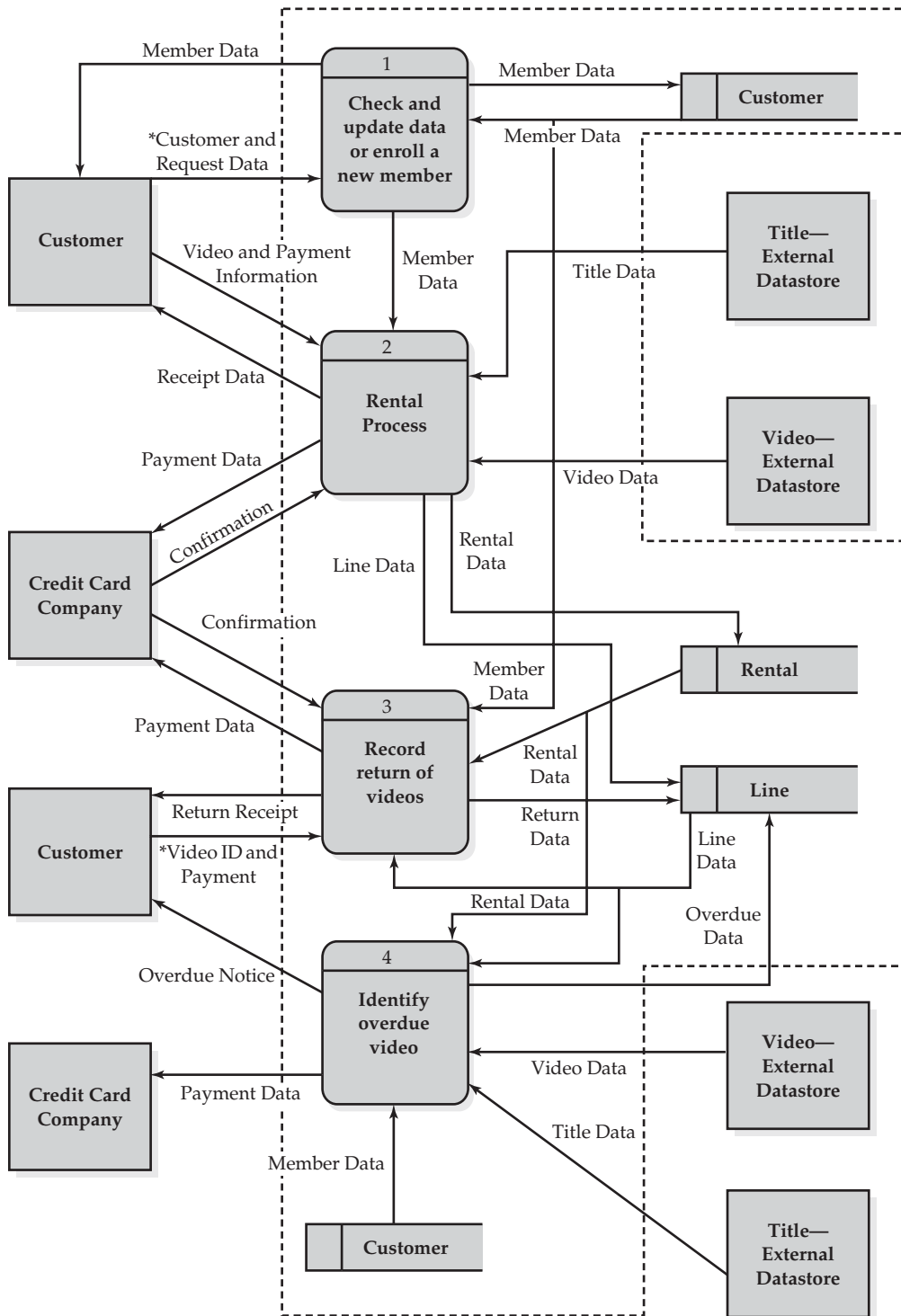
FIGURE 8.3
Context-Level
DFD for the
GB Video
Proposed
System



projects in organizations can contain more. During preparation of the narrative, the GB team identified four major processes: member, rental, return, and overdue. These processes must appear on the first explosion to keep the narrative and MDFDs consistent. Since member and rental are sequential processes, the two could be combined into a single major process in both the narrative and MDFDs. The return and overdue processes occur at different times with different triggers and thus cannot be combined with member and rental processes.

The first explosion MDFD shown in Figure 8.4 corresponds to the description in the narrative model and follows MDFD rules. Every process in the narrative appears on the MDFD (unless it is described as part of a separate system) and every process on the MDFD appears in the narrative. Because the rental process contains the word process in the name, rental process will have an explosion. Every data store corresponds to a store mentioned in the narrative and matches (or will match) one of the entities on the CDM. Every data flow is mentioned in the narrative and/or must be present to satisfy the DFD completeness rules.

FIGURE 8.4 First Explosion MDFD for the GB Video Proposed System



Each of the four processes has a trigger. The member and return processes have external data flow triggers; the triggering data flows from the externals display an * in the flow name. Because the proposed system design calls for access to the rental process only from the member process, the only way to trigger the rental process is with a data flow from the member process. This design should prevent nonmembers from renting tapes. The overdue process does not connect to a process-to-process flow or to a trigger flow from an external. Process 4 is a batch process triggered by time, that is, a clerk triggers it after all the overnight returns are processed. The diagram could show a dotted line control flow from an external called System to the overdue process, if desired.

Additional Explosions

Careful, well-structured design at the specifications level can greatly simplify coding at the next stage. When a number of different functions take place inside a process on the first level explosion, the analyst may choose to explode the process further for clarity. The member, rental, and return processes all have several different functions inside and are candidates for further explosion. The analyst must use judgment to determine the level of detail that provides the most clarity. Too little detail may obscure important functions. Too much detail will add little to clarity and may confuse the viewer.

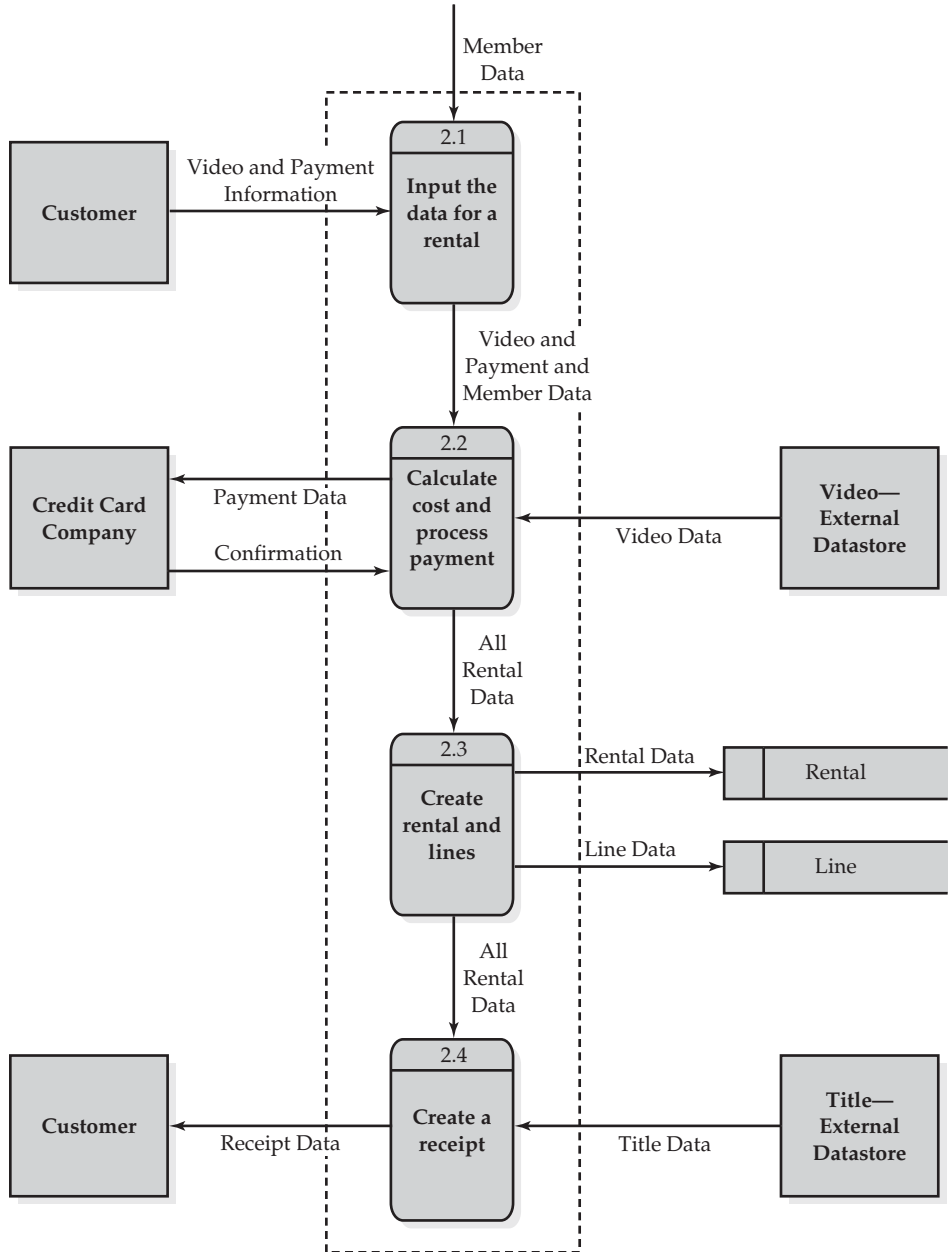
An illustrative explosion for process 2 appears in Figure 8.5. In this example, Rental Process 2 becomes four subprocesses. Following good practice, each subprocess performs only one major function: 2.1 enters input data, 2.2 receives payment; 2.3 creates new records in two data stores, and 2.4 prepares the customer receipt. One could explode process 2 into more than four subprocesses or could explode one or more of the subprocesses. For example, 2.4 could explode to 2.4.1 Retrieve Video and Title data and 2.4.2 Generate receipt.

Explosions for MDFDs follow the rules for DFD explosions. For example, the consistency or balance rule requires that every flow connecting to a process must appear and be resolved on the explosion. Note that the sample explosion is balanced. All of and only the flows to and from process 2 on the first explosion diagram appear as external flows on the diagram for the explosion of process 2. Three new process-to-process flows appear: from 2.1 to 2.2, from 2.2 to 2.3, and from 2.3 to 2.4. These flows cannot appear on the first level explosion because they connect subprocesses that exist only on the second explosion. Under MDFD rules, these flows indicate that process 2.1 triggers 2.2; 2.2 triggers 2.3; and 2.3 triggers 2.4. These processes occur in sequence and immediately after one another. The explosion also shows that 2.1 is the subprocess part of rental process 2 that is triggered by a data flow from process 1.

GRAPHICAL DATA SPECIFICATIONS

The conceptual data model defines the data specifications for the proposed system without reference to the physical implementation. The CDM includes entities, attributes, relationships, minimum cardinalities, associative and weak entities, and

FIGURE 8.5
Explosion of
Process 2



subtypes/super types as appropriate. The CDM defines the content of the data stores on MDFDs. Each data store on the MDFDs is defined by one and only one entity. Alternatively, the team may choose to represent the specifications for both data and process in an object-oriented model.

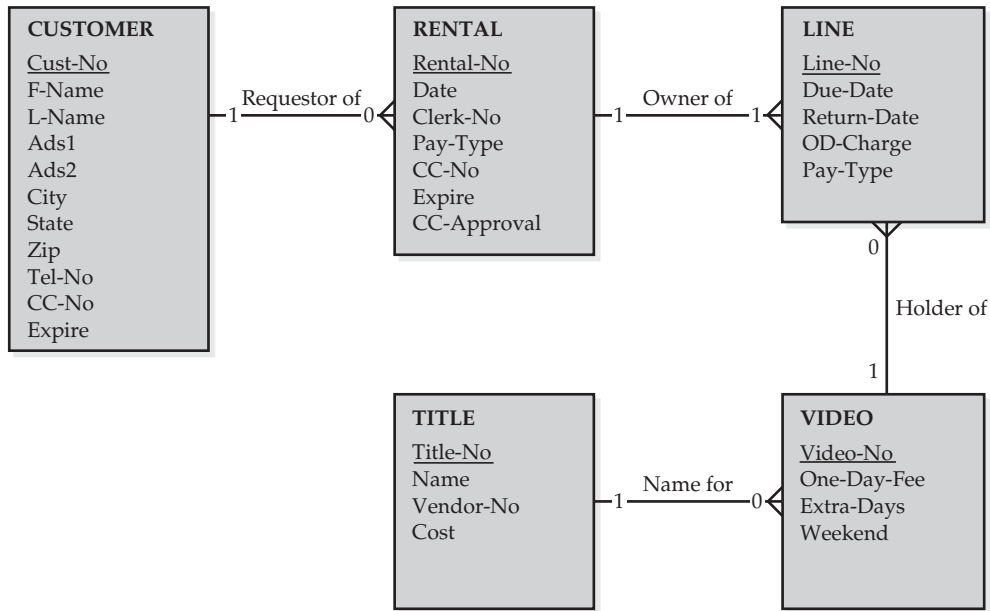
The team can construct the CDM using top down analysis: What are the things (entities) about which the new system will collect and use data? The EDM for the current operation may help the team to identify the strong entities; however, the data stores in the new system may contain data from strong entities that are not part of the current operation and may omit ones that are. The CDM also must match the narrative model. As the design proceeds, the team may need to revise the narrative, DFDs, and/or CDM to keep them consistent.

Once the entities are identified, the team can proceed to add relationships, cardinality, attributes, and primary identifiers. Examine carefully the functions that the client wants the new system to perform and make sure that the team has the entities, attributes and relationships needed to perform the functions. In some cases, the team may discover super-type/sub-type structures. The CDM should reflect these when they exist.

Guidelines for a proposed system CDM are as follows:

1. Select and use a standard notation. State on the diagram if any unusual notation is used.
2. Include all of the entities that define: (a) data stores within the bounds of the new system and (b) external data stores that are used by the new system.
3. Add the attributes of each entity to define the data store contents. A data store contains all of and only the attributes of the defining entity.
4. Show a primary key or identifier in standard notation for every entity. If the primary identifier of an associative entity is the composite of the primary identifiers of the linked entities, it need not appear.
5. Include the relationships with both maximum and minimum cardinalities correctly represented.
6. Replace many-to-many relationships with associative entities.
7. Convert multivalued attributes to entities.
8. Eliminate as much data duplication as possible. Convert reference data that is common to multiple instances of an entity, for example, the title data for a video, into entities.
9. The presence of foreign keys implies that the team has selected a logical data model, that is, the relational model. For this reason, foreign keys should not appear on the conceptual data model.
10. Diagrams must appear clean, neat, clear, and readable.

A CDM for the proposed GB Video computer-based system appears in Figure 8.6. The simplified, reduced form notation used in the diagram is explained in Chapter 4 and is described on the diagram. The associative entity, Line, must appear to resolve the m:n relationship between Rental and Video. Information that is common to several Video instances appears in a new

FIGURE 8.6 Conceptual Data Model for GB Video

Note: the diagram uses Simplified, reduced form notation. Attributes are placed inside the entity boxes and relationship diamonds are omitted.

reference entity, Title. The diagram shows the entity Rental with the attributes of credit card number (CC-No) and the credit card expiration date (Expire) because customers may pay with any credit card, not just the one they used to become a member. The narrative also says that most customers pay cash. If desired, the team could use a super-type entity called Rental with a subtype entity called CC Rental.

All of the content on the CDM must appear either directly or indirectly in the narrative model and vice versa. Note that two entities from the current operation EDM, Employee and Vendor, do not appear in the CDM because the proposed system specifications do not include retrievals from or storage of Vendor or Employee data. Because of the ERD/DFD integration rule for MDFDs, the data stores on MDFDs for the proposed system must correspond exactly to the entities in the CDM. Two of the entities, Title and Video, appear as external data stores.

METADATA SPECIFICATIONS

Metadata for the requirements specification consist of natural language text to describe the objects that make up the models of the new system. At the requirements specification level, the descriptions focus on the functional roles

or logic of the objects. Data objects include entities, attributes, and relationships. The metadata for them consist of descriptions for each of the objects. Process objects include externals, processes, data stores, data items, and data flows. The team may reuse descriptions from the current operation metadata when appropriate.

1. Externals, metadata consist of descriptions.
2. Data store metadata may consist of the definition of the store, the attributes contained within it, plus the size or expected number of records, retention policy for records, and access control policies.
3. Data flow metadata consist primarily of the contents of the flow—the attributes or data items within the flow. If the flow contains data items, data elements that are not attributes, then each data item should have a description.
4. Process metadata provide the full logic for each process in text form, such as what triggers it, any such data operations as retrieve or store, what happens in the process, and what other processes it triggers. Every DFD process should have a logic description. Normally, the metadata for a process come directly from the narrative with editing as appropriate.

Figure 8.7 shows sample metadata for the GB Video new system.

OBJECT-ORIENTED DESIGN SPECIFICATIONS

The content model discussed in Chapter 1 defined information systems as containing data, process, and infrastructure. At the conceptual model level, both the object model and process model frameworks specify the same data and process content of an information system but use different representations. Each representation highlights certain aspects and structure of the target information system.

With a process model framework, the analyst represents the specifications as structured around the major processes. The DFD process model shows the processes and also shows how each process links to or operates on data. With modified data flow diagrams (MDFDs), the conceptual data model (CDM) defines the content of each data store. As shown in Figure 8.4, the GB Video proposed system contains processes for (1) enrolling a member, (2) renting a video, (3) recording the return of a video, and (4) identifying overdue videos. The CDM in Figure 8.6 defines the data store contents for the GB Video proposed system.

With object-oriented design, the analyst represents the specifications as structured around objects, not around processes. The objects contain within them both the data and operations (processes) required to carry out their desired behavior. The link between data and processes occurs within the objects and in interactions between objects. The structure of the conceptual-level object model resembles the structure of the conceptual data model. For the GB Video proposed system, the conceptual-level object classes are (1) customer, (2) rental, (3) rental-line, and (4) video, a quite different structure for the object model than for the

FIGURE 8.7 Metadata for the GB Video Proposed System

Entities, Attributes, and Relationships for the CDM. The tables below contain the entity, attribute, and relationship data for the CDM of GB Video.

Entity	Description
CUSTOMER	Contains all the available information about each customer who has made a transaction in the last year
LINE	Contains the information on each video associated with a rental transaction
RENTAL	Contains the information on each rental transaction
TITLE	Contains information on each distinct title of the videos
VIDEO	Contains information on each individual video

CUSTOMER	
Attribute	Description
Cust-No	A unique identifier assigned to each member
F-Name	First name and middle initial if any
L-Name	Last name
Ads1	Street or box address
Ads2	Apartment number or other as needed
City	Name of city
State	State id code
Zip	Zip code
Tel-No	Telephone number
CC-No	Credit card number
Expire	Expiration date on the credit card

RENTAL	
Attribute	Description
Rental-No	Unique identifier assigned to each rental
Date	Date of the rental
Clerk-No	Employee number of the clerk entering the rental
Pay-Type	Cash, check, or credit card
CC-No	Credit card number
Expire	Expiration date of the credit card
CC-Approval	Credit card approval code

LINE	
Attribute	Description
Line-No	Unique identifier assigned to each line
Due-Date	Date tape is to be returned
Return-Date	Actual return date
OD-Charge	Charge for days kept after due date, if applicable
Pay-Type	Method of payment for the overdue charge

VIDEO	
Attribute	Description
Video-No	A unique identifier assigned to each videotape
One-Day-Fee	First day rental fee
Extra-Days	Extra days rental fee
Weekend	Rental fee for Sat. and Sun.

TITLE	
Attribute	Description
Title-No	Unique identifier for the title
Name	The name of the video, e.g., <i>Charlie's Angels</i>
Vendor-No	Identifier for the vendor who produced/sold the video
Cost	Purchase price for the video

Relationship Name	Description	Entity1 with Min, Max Cardinality	Entity2 with Min, Max Cardinality
Requestor of	Links each customer to rentals made by the customer	CUSTOMER—a rental <i>must</i> be for one customer (1, 1)	RENTAL—a customer <i>may</i> make many rentals (0, many)
Owner of	Links each rental to the associative entity	RENTAL—a line <i>must</i> belong to one rental (1, 1)	LINE—a rental <i>must</i> contain one or many lines (1, many)
Holder of	Links the associative entity to a specific video	LINE—a video <i>may</i> be held by many lines (0, many)	VIDEO—a line <i>must</i> hold one video (1, 1)
Name for	Links a title to the videos that use the title	VIDEO—a title <i>may</i> be the name for many video (0, many)	TITLE—a video <i>must</i> be named by one title (1, 1)

MDFD Data Stores

Each data store is defined by the entity in the CDM with the corresponding name.

Externals

Name	Description
Customer	People who rent and return videos from GB Video
Credit Card Company	The processing center that receives a requested transaction on a credit card number, verifies it, and issues an approval code

Data Flows

1. All Rental Data: the attributes of Video, Customer, Rental, and Line plus rental-subtotal, tax, and total-rental-cost
2. Confirmation: CC-No, CC-Approval
3. Customer and Request Data: the attributes of Customer
4. Line Data: the attributes of Line
5. Member Data: the attributes of the entity Customer
6. Overdue Data: Line-No, OD-Charge, Pay-Type
7. Overdue Notice: the attributes of Customer and Title plus Due-Date, OD-Charge, Pay-Type, CC-No
8. Payment Data: CC-No + Total-charge
9. Receipt Data: All Rental Data + Name
10. Rental Data: the attributes of Rental
11. Return Data: Rental-No + Video-No + Return-Date + OD-Charge + Pay-Type
12. Return Receipt: Member Data + Rental Data + Line Data
13. Title Data: the attributes of Title
14. Video and Payment and Member Data: Video and Payment Information + Member Data
15. Video and Payment Information: Video-No + Payment-Type + CC-No + Expire
16. Video Data: the attributes of Video
17. Video ID and Payment: Video-No, Pay-Type

Processes

1. The member process is triggered by a customer request to (1) rent videos and/or (2) become a member. If the customer has and knows the customer number, the number is entered; the system retrieves the record from the Customer data store and displays the customer data. If the customer does not have the number, the customer can provide a telephone number (or a name and zip code). The system tries to retrieve the customer's record from the Customer data store.

If the system is unable to retrieve the customer record or if the customer is not a member, the new member subprocess is initiated. The system will create a new member provided the person has a credit card, telephone, and government-issued ID. The customer supplies the customer data and the data are entered. The system generates a customer number, creates a membership card output, and gives the output to the new member. The system creates a record in the Customer data store for the new member.

Once the appropriate customer record is available, the customer is asked (1) to verify the name, address telephone, and credit card data and (2) to report any changes or corrections. This subprocess increases the likelihood that the system will contain current information for the customer. Any change data are entered and the customer data store is updated. When a verified customer record is available and the customer wishes to rent, the member process triggers the rental option and the member data are sent to the rental process. The rental process can be accessed only from the member process; the rental process cannot be accessed directly.

2. The rental process is triggered by and only by the member process. The process begins with 2.1.

2.1 This process accepts the member data from the member process and generates a new rental transaction number and the rental date. The customer provides the video number and the proposed return date, the due date. The video number and the due date are entered into the system. The system retrieves the video data from the Video data store and, based on the due date, calculates the rental price for each video rental. This process is repeated for each video. After all the videos are processed, process 2.2 is triggered.

2.2 The system adds the rental price for each video to get a rental subtotal. The system multiplies the rental subtotal by the tax rate (state sales tax + county sales tax + city sales tax) to get the tax. The total rental cost is the rental subtotal plus the tax. The payment type—cash, check, or credit card, is supplied by the customer and entered into the system. If payment is by credit card, the credit card data are entered. The system sends the credit card data and the total rental cost to the credit card processor. If the transaction is approved (or is for cash or a check), the process triggers 2.3.

2.3 This process “commits” the rental and line records, that is, the records are created in the data stores. The rental number, date, clerk number, pay type, and credit card data go to the rental record. The due dates go to the line records for each video. The process triggers 2.4.

2.4 This process generates a receipt for the customer. The receipt data includes all of the rental data supplied by the customer, sent from the member process, and generated by the system during the rental transaction. The system also retrieves the title data from the Title data store and includes the name in the receipt data. The rental process ends here.

3. When a video is returned, the video number is entered into the system. The system retrieves the corresponding line and rental records, enters the return date, and calculates overdue charges if any. The customer may charge the overdue fee to the credit card used for the rental or may pay by cash or check. About 95 percent of the time, customers return videos with no payment type input, for example, drop them in a return box. In the absence of customer input on payment, the system retrieves a credit card number from the customer record and processes the overdue charge against the credit card. The system records the charge and payment type in the Line data store. The system may create a return receipt for the customer. The return process ends.

4. After the overnight returns are processed, a clerk instructs the system to run the overdue program, that is, it triggers the overdue function. The system processes the Line records. For each video that is two or more days overdue (today's date – due date ≥ 2), the system retrieves the rental record for the video, retrieves the customer record for the rental, generates an overdue notice, and sends the notice to the customer. When a video is 14 days overdue, the system retrieves the customer's credit card number from the Customer data store and the video cost from the Video and Title data stores, charges the customer's credit card for the amount of the cost of the video, and sends a notice informing the customer of the charge. The overdue process ends. If a customer has a complaint about overdue charges, the complaint is handled and processed by Accounting.

Page 5

process model. An analyst preparing a conceptual-level OOD model for a system probably will prepare a Use Case Diagram and a Class Diagram. The analyst may create other OOD diagrams during detail design when the build option is selected.

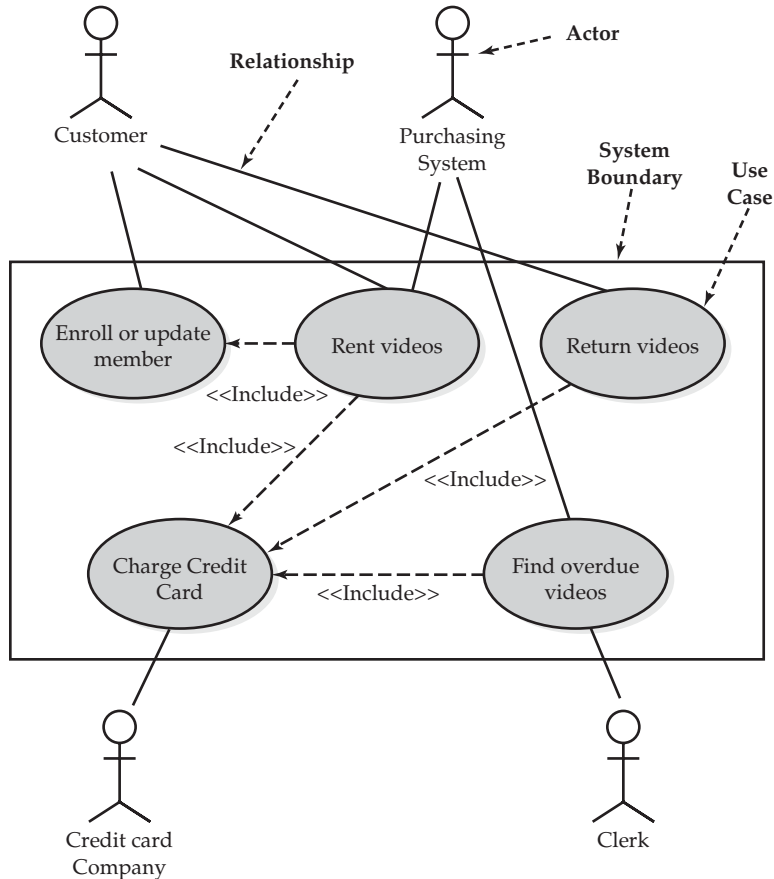
Use Case Diagrams

Use case diagrams, process models that resemble DFDs, give the analyst a way to specify the interaction of system processes with each other and the external environment. Figure 8.8 shows the Use Case Diagram for the GB Video Proposed System.

The use cases on the diagram correspond to the processes in the GB Video first explosion MDFD in Figure 8.4. Three of the use cases, Rent videos, Return videos, and Find overdue videos, all involve a common activity: they contact the credit card company and place a charge against the customer's credit card. The use case diagram represents this situation by adding a use case called Charge credit card and letting the other three use cases "include" the Charge credit card use case. Rent videos also includes the Enroll member use case. The Rent and Find use cases interact with the purchasing system in that they use data from the Video and Title data stores maintained by Purchasing. With OOD modeling, all of the behaviors shown on the use case diagram must appear as behaviors associated with one of the objects in the system. Unfortunately, the use case diagram provides no guidance as to which behavior goes in which object.

The metadata or scenario for each of the use cases forms an important part of the use case diagram. Figure 8.9 shows the scenario for the Rent videos use case in Figure 8.8. This scenario contains some header information, including a Precondition and a Trigger. The Precondition tells what the system must ensure is true before this use case begins. The Trigger specifies the event that

FIGURE 8.8
GB Video
Proposed
System Use
Case Diagram



starts the operation of the use case. The Main Success Scenario describes the things that happen when all goes well and the end result is a successful rental. The Extensions show by line number in the Main Success Scenario, things that can go wrong and prevent a successful rental. The underline in steps 3 and 7 indicates that these steps include another use case as shown in Figure 8.8.

Class Diagrams

A **Class Diagram** specifies each object class in a system, identifies the data and operations for each object in the object class and shows the associations between objects. A simplified conceptual-level class diagram for the GB Video proposed system appears in Figure 8.10. The diagram omits some of the operations related to the rental return system, for example, the operations pertaining to checking and charging the customer's credit card and refusing rentals for nonmembers.

No clear rules exist on how to select the objects for the class diagram. In general, the entities that appear in the conceptual data model become the objects for the class diagram. In Figure 8.10, the entities from the CDM in

FIGURE 8.9 Scenario for the Rent Videos Use Case

Rent Videos Scenario

Precondition: None

Trigger: Customer decides to rent one or more videos.

Main Success Scenario

1. Customer selects the videos to rent.
2. Customer goes to checkout and provides member data.
3. System confirms that customer is a member and retrieves and updates member data.
4. Customer provides the video number for each video.
5. System retrieves charge, due date, and title data and creates a rental with a line entry for each video.
6. System calculates the total charge.
7. System contacts the credit card company and enters the total charge.
8. System produces a receipt for the customer.

Extensions

- 3a. The customer is not a member.
 1. Sign up the customer as a member.
 2. Else, terminate rental.
- 7a. Credit card company declines to authorize purchase.
 1. Ask customer for an alternate card.
 2. Else, terminate rental.

FIGURE 8.10
A Conceptual-Level Class Diagram for the GB Video Proposed System.

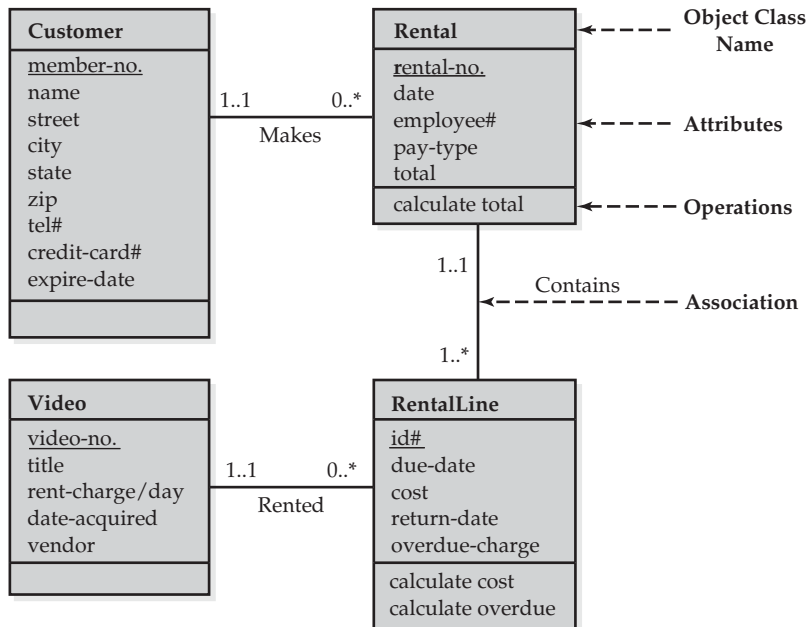


Figure 8.6 appear as objects except for Title. Title has no function except to provide data, and in Figure 8.10, the Title data are part of the Video objects. Video appears as an object class on this diagram because Video objects play an important role in the rental system. The Video object class is controlled by the purchasing system. Other objects in the rental system can ask the video objects to retrieve data but may not ask the video objects to perform any other operations.

The use case diagram for GB Video in Figure 8.8 contains these four use cases:

1. Enroll or update member.
2. Rent videos.
3. Return videos.
4. Find overdue videos.

Each of these behaviors must be contained in or associated with an object class. The analyst must decide which object is associated with which behavior.

The “Enroll or update member” behavior clearly seems associated with the Customer object class. However, the only operations associated with this behavior are to retrieve member data, create a new customer object for each new member, update member data, and delete the appropriate customer object when a member becomes inactive. Since all object classes contain the CRUD operations of create, update, retrieve, and delete, the CRUD operations are not shown in the operations area of the object class. As a result, the Customer object class contains no entries in the Operations area.

The “Rent video” behavior results in creating a Rental object and one or more Rental Line objects. The create operations are assumed and not shown in the operations areas for the Rental and Rental Line object classes. However, the Rental object class shows additional behavior to calculate the value for a derived attribute of Rental called “total,” the total cost of a rental. A rental object must obtain from each associated rental-line object the data on the cost, add the costs to obtain a subtotal, and calculate and add the sales tax to get a total cost for the rental object. This behavior appears in Figure 8.10 as an operation called “calculate total” in the Rental object class. In similar fashion, the RentalLine object class has an operation to calculate the derived attribute, cost, perhaps an operation that multiplies the number of days the video is rented by the cost per day. The “Rent video” behavior also results in the retrieval of data from the associated video objects. The Video object class may contain other operations, but the other operations are not available to the Rental system and are not shown on the class diagram.

The “Return video” behavior updates data in the associated RentalLine object. Returning a video has no effect on the Rental object, only on the RentalLine object for the video that is returned. The RentalLine object class also contains an operation to calculate the derived attribute, overdue-charge, when a video is returned after the due date. The “Find overdue videos” behavior is associated with the RentalLine object class because the data on due-date appears in the RentalLine objects. The overdue videos are identified by the operation “calculate overdue.” This operation displays polymorphism or different behaviors depending on the situation. “Calculate overdue” finds the overdue charge, if any, when a video is returned and also finds all of the overdue videos during the daily processing cycle.

The OOD version of the GB Video proposed system as represented by the conceptual class diagram contains all the same data and behavior as the process model representation but uses a very different structure. Some analysts prefer the OOD representation especially when the system will be built with an object programming language.

Summary

Requirements specification for the proposed system marks the beginning of the synthesis phase of a project. *Synthesis* means combining separate pieces to form a coherent whole, that is, a proposed system that meets the client's goals. During requirements specification, the team begins to assemble the pieces for the proposed system. Imagination, creativity, knowledge, and experience all play important roles during synthesis.

The goal of the proposed system phase is to assure that the conceptual specifications for the proposed system are complete and correct before the team expends any significant effort on the logical and physical design—a philosophy of “make it right; then make it work.” Conceptual specifications represent solutions that a team can implement in different sourcing options and in different technology environments. Developing a complete and correct understanding of the conceptual specifications of the system can consume substantial time and effort and may involve additional information gathering activities. During this phase, the team creates answers to questions that include (1) what functions or processes should the system include to provide the features that the client wants and (2) what data are required to support or operate the included functions.

The team may answer these questions by preparing such major deliverables containing the proposed system specifications as:

- *Narrative specifications for the proposed system.* The narrative model specifies the proposed system in natural language and follows a specific format to encourage completeness and facilitate communication between team members.
- *Graphical process specifications.* Many projects will create the conceptual process model with modified data flow diagrams (MDFDs); however, other process models may work better for some projects.
- *Graphical data specifications.* A conceptual data model (CDM) includes all of the entities that will define data stores in the new system.

The team must solve a number of problems to determine specifications. The problem-solving methods discussed here apply to all the aspects of the synthesis phase: proposed system specifications, creating and evaluating alternatives, outsourcing, and system design. Most problem-solving methods fall into one of several classes:

- *Experience-based methods* that use prior experience or experiences of other organizations to derive solutions.
- *Trial and error methods* that try out different solutions.
- *Heuristic methods* that follow a set of rules that seem to work well but do not guarantee a best solution to a problem.
- *Difference reduction methods* that identify and try to reduce the differences between the current state and the desired state.

- *Calculation and optimization methods* that use mathematical procedures to find a solution.

The value chain model of an organization also can provide guidance on developing specifications for the proposed system. A value chain represents the value production function for the organization: the activities that result in the total value, net benefits, or profits produced by the organization. The analyst may use a combination of several problem-solving methods to solve the system synthesis problems.

Modified data flow diagrams (MDFDs) often provide a good process model at the conceptual level. MDFDs include all the features and rules of DFDs plus some rules that help the analyst to focus more clearly on specifications for the proposed system. The additional rules for MDFDs include:

- DFD/ERD integration. *All of the entities and only the entities on the conceptual ERD for the system appear as data stores in MDFDs.*
- Process triggers. *Every process on the MDFDs must be triggered by a data or control flow.*
- Time and immediacy. *A data flow from process A to process B always triggers process B.*

The CDM defines the data specifications for the proposed system without reference to the physical implementation. The CDM can consist of a context-level diagram, a first explosion and additional explosions as appropriate. The CDM includes entities, attributes, relationships, minimum cardinalities, associative and weak entities, and subtypes/super types as appropriate. The CDM defines the content of the data stores on the DFDs. Each data store on the DFDs is defined by one and only one entity.

Metadata for requirements specification consist of natural language text to describe the objects that make up the models of the new system. At the requirements specification level, the descriptions focus on the functional roles or logic of the objects. Data objects include entities, attributes, and relationships. Process objects include externals, processes, data stores, data items, and data flows. Metadata often is presented in a table format.

Alternatively, the team may choose to represent the structure of both data and process in an object-oriented model. With object-oriented design (OOD), the analyst represents the specifications as structured around objects not around processes. The objects contain within them both the data and operations (processes) required to carry out their desired behavior. The link between data and processes occurs within the objects and in interactions between objects. The structure of the conceptual-level object model resembles the structure of the conceptual data model.

A class diagram specifies each object class in a system, identifies the data and operations for each object in the object class, and shows the associations between objects. No clear rules exist on how to select the objects for the class diagram. In general, the entities that appear in the conceptual data model become the objects for the conceptual-level class diagram.

Use case diagrams, process models that resemble DFDs, give the analyst a way to specify the interaction of system processes with each other and the external environment. The metadata or scenario for each of the use cases forms an important part of the use case diagram. With OOD modeling, all of the behaviors shown on the use case diagram must appear as behaviors associated with one of the objects in the system. Unfortunately, the use case diagram provides no guidance on which behavior goes in which object. The analyst must decide which object is associated with which behavior.

The OOD graphical specifications for a proposed system as represented by the conceptual class diagram contain all the same data and behavior as the process model specifications but use a very different structure. Some analysts prefer the OOD representation especially when the system will be built with an object programming language.

Key Terms

automate, 261	discover, 261	narrative specifications, 266
best practices, 260	experience-based methods, 259	object-oriented design (OOD), 258
brainstorming, 259	graphical model specifications, 267	optimization, 263
calculation and optimization methods, 259	heuristic methods, 259	optional process, 268
class diagram, 288	immediacy, 274	problem solving, 259
conceptual data model (CDM), 258	inform, 261	proposed system specifications, 258
conceptual specifications, 256	mandatory process, 268	synthesis, 256
DFD/ERD integration, 274	metadata, 281	transform, 261
difference reduction methods, 259	modification of the current system, 265	trial and error methods, 259
	modified data flow diagram (MDFD), 258	trigger, 269
		use case diagram, 287
		value chain model, 263

Review Questions

Answer the following questions regarding these topics.

- Proposed system specification.
 - Explain why proposed system analysis is synthesis rather than decomposition.
 - What does it mean to say that proposed system analysis should be conceptual?
- Proposed system outcomes.
 - What is the goal of the proposed system phase?
 - Who are the audiences for the different components of the proposed system specification?
- Problem solving.
 - Explain the five different problem-solving methods discussed in the chapter.
 - Give an example of when you would use each of these methods.
 - What are the four basic functions that a computer system can perform to produce a difference reduction outcome?

4. Narrative models.
 - a. How is the narrative model for the proposed system different from that of the current system?
 - b. Who is the audience for the narrative model?
 - c. What is the difference between a mandatory and a desirable process? Give an example of each.
5. Graphical process models.
 - a. What must the graphical process model do?
 - b. What does a modified data flow diagram include that an ordinary data flow diagram does not include?
 - c. What is a trigger? Give examples.
 - d. How is a trigger indicated on an MDFD?
 - e. What denotes a data flow that goes to more than one process on the first explosion?
6. Conceptual data models.
 - a. What is a conceptual data model?
 - b. How does a CDM handle keys and foreign keys?
 - c. How does a CDM handle a many-to-many relationship?
7. Model integration.
 - a. What do the narrative model and the graphical process model have in common?
 - b. What do the graphical data model and the graphical process model have in common?
8. Object-oriented models.
 - a. How does an object-oriented model differ from an MDFD/CDM model?
 - b. What are three basic diagrams used to implement an OO model?
 - c. What is the purpose of each of these diagrams?
9. Team responsibilities.
 - a. Should the team design the system around the existing technology? Explain your answer.
 - b. Should the team create options and suggestions or let the clients decide what they want without any team input? Explain your answer.
 - c. What are some issues that a team might include in an organizational model?

Critical Thinking Exercises

Individual Exercises

1. You have been asked to develop a new system for a small library to track books. The library has three computers that are used for such basic tasks as typing letters and sending e-mails.
 - a. Would you use the modification approach or plan a new design? Explain your answer.
 - b. In the library situation, which problem-solving methods will you use initially to develop specifications for a proposed system? Justify your answer.
2. Describe each stage in the value chain of each of the following organizations:
 - a. A manufacturer that produces a product.
 - b. A store that sells products at retail.
 - c. A service firm such as a doctor's office or a law firm.
 - d. A public service organization such as a police force.

Group Exercises

1. The library from Individual Exercise 1 works as follows. The customer selects a book and brings it to the librarian, along with the customer's library card. The librarian makes a copy of the library card along with the book information and files this information in a folder under the date. When the customer returns the book, the librarian removes this information from the folder.
 - a. What processes might be included in a narrative model for the library book loan system?
 - b. Choose one of these processes and explain it in narrative model format.
 - c. Define the metadata for the process you explained in problem (b.) above.
2. Draw a context and first level MDFD for the library.
3. Redraw the class diagram for GB Video in the textbook to serve as a class diagram for the library problem.

References

- Hoffer, Jeffrey A.; Joey F. George; and Joseph S. Valacich. *Modern Systems Analysis and Design*, 4th ed. Upper Saddle River, NJ: Prentice Hall, 2005a.
- Hoffer, Jeffrey A.; M. B. Prescott; and F. R. McFadden. *Modern Database Management*, 7th ed. Upper Saddle River, NJ: Prentice Hall, 2005b.
- Porter, Michael E. *Competitive Advantage: Creating and Sustaining Superior Performance*. New York: Free Press, 1985.
- Post, Gerald V. *Database Management Systems*, 3rd ed. New York: McGraw-Hill/Irwin, 2005.
- Whitten, Jeffrey L.; Lonnie D. Bentley; and Kevin C. Dittman. *Systems Analysis and Design Methods*. New York: McGraw-Hill/Irwin, 2005.
- Zuboff, Shoshana. *In the Age of the Smart Machine: The Future of Work and Power*. New York: Basic Books, 1988.

Chapter Nine

Alternatives, Evaluation, and Recommendation

Chapter outline

Introduction

Making Choices

Alternative Solutions

Choosing a Design Option

Choosing Functionality

Choosing a Sourcing Option

Building a Solution In-house

Outsourcing the Solution

Choosing Infrastructure

Evaluating Performance

Describing Alternative Solutions

Evaluation

Feasibility

Risk Analysis

Cost/Benefit Analysis

Benefits

Costs

Evaluation Metrics

The Implied Benefits Method

Cost/Benefit Table

Features Analysis

An Example of Alternatives

The Evaluation Comparison and
the Recommendation

The Recommendation

Client Approval to Proceed

Summary

Key Terms

Review Questions

Critical Thinking Exercises

Individual Exercises

Group Exercises

References

INTRODUCTION

Evaluation and alternatives remain central themes throughout the life cycle of a system. The team begins the evaluation process with the analysis of strategic alignment during project definition. The team also generates acceptable sourcing options as part of project definition. At the beginning of the Proposed System Phase, the team prepares the conceptual specifications for the proposed system. Once the conceptual specifications for the proposed system are known, the team may identify and evaluate a number of alternatives to find the systems that appear to provide satisfactory solutions or the recommended system(s). Good evaluation involves close interaction with the client. The team can and should

clarify and quantify the choices, but only the client can determine how well a recommendation meets the organization's or the sponsor's values, goals, and objectives. The recommendation represents a key decision point. When the client accepts the recommendation, the team enters into the system delivery phase to procure a solution from a vendor, build a system, or some combination.

A recommended solution involves two interrelated choices: (1) the specification of features—the levels of functionality, infrastructure, and performance included in the system; and (2) the sourcing path—the specific build or buy option. The first choice requires the organization to look at the cost-effectiveness or value impact of each feature in the proposed specifications for the new system. If the cost of implementing the full set of specifications appears to exceed the value or the client's budget constraint, the organization will need to remove some of the less cost-effective features from the specifications. The second choice, the sourcing path, may further complicate the decision. An outsourced solution with a desirable total cost may not offer all of the features that the organization wants and a custom-built internal solution with all the desired features may fail the cost-effectiveness test. Evaluation remains important during system delivery. A team that decides to outsource will use evaluation techniques to select the most suitable procurement option, for example, to select the most suitable package from among the products of various vendors. A team and client that decide to build may use evaluation techniques to determine which features to include especially when the recommended system appears to cost too much or take too long to build.

The initial evaluation decisions about functionality, performance, infrastructure, and sourcing that lead to a recommendation occur before the system delivery work when the team knows less of the relevant information. As additional information on cost, risk, and available vendor options unfolds, the team and client revisit the earlier choices as shown clearly in the Spiral Model. The client may relax some requirements to save money or add additional features if the costs are acceptable. A client who initially wants to build the proposed system may decide to buy and install a package, hire a contractor to build the system, outsource the entire function to a vendor, or some combination. A client who started with an outsource preference may switch to a build option if the choices available from vendors omit mandatory requirements or cost too much. Sometimes the client may cancel or substantially revise the entire project. While major changes can and do occur at any point in the solution process as the team and client continue to learn more about the problem, change becomes increasingly expensive and potentially disruptive. Major changes during system delivery may result in losing the benefits of some or most of the possibly large expenditures to date of time and monies for design, programming, testing, and/or vendor products.

The most accurate evaluation probably occurs after system delivery. Many clients conduct a *post implementation audit*, an evaluation to determine if the system actually operates as planned and/or produces the expected benefits. Additional evaluation may occur after an extended period of use to determine whether modification is feasible and desirable or whether the system has reached the shutdown point. Thus, evaluation continues from project definition until the final shutdown of the system.

The input to and output from the alternatives, evaluation, and recommendation phase of the proposed system include:

- *Input.* Project definition and conceptual specifications for the proposed system.
- *Output.* The recommended solution.

For the recommended solution, the team explores and analyzes the evaluation issues in as much depth as practical, makes best guesses or estimates as necessary, and chooses a **recommendation**. With client agreement, the team then proceeds to system delivery for (1) a buy option, as we discuss in Chapter 10 or (2) a build option, as we discuss in Chapter 11. In a few cases, the client may wish for the team to explore both build and buy options in depth. The team might build a prototype that contains the exact functions the client wants. In addition, the team may find packages that match as closely as possible with the client's requirements. The team can demonstrate both the prototype and the packages and let the client make a final buy/build choice. However, in many cases the cost for multiple alternatives is too high. Evaluation, along with most things in systems work, involves compromise and good judgment.

MAKING CHOICES

The client often will present a preferred solution as part of the goals for the new system. The team should look carefully at the client's preferred choice, but the team also should exercise professional responsibility. The client may lack an IT background and/or may not have studied the problem in depth. Senior managers sometimes catch the "airline magazine" syndrome, that is, they read about an alternative in an airline or similar glossy magazine and decide that their organization needs it. The team's professional obligation requires the team to explore a reasonable set of alternatives and present an evaluation and recommendation to the client. The team always wants to seek out and understand client preferences, but most clients will appreciate information contrary to their views or preferences especially when it is presented in a nonconfrontational manner.

Usually the team can identify several, often three or four or more, reasonable alternatives for a new system. The next step is to evaluate the alternatives and choose a recommended one. While people often talk about finding the "best alternative," in practice, the team will find great difficulty identifying what the best alternative even means let alone how to find it. "Finding the best alternative" really means avoiding the clearly unsatisfactory alternatives and finding one or more satisfactory alternatives (March and Simon, 1958). To choose a recommended alternative, the team considers such factors as the organization's strategy, costs and benefits, risks, and available resources. The team conducts the evaluation within the framework of the culture and values of the organization. The team identified these values during the strategic alignment step of project definition. The team also identified the scope and possible solutions that are acceptable to the client. Culture and values influence many of the trade-off decisions. For example, some clients lean strongly toward low-risk solutions while

others want to be “leaders,” even if leadership involves higher risk. Some clients prefer to build all applications while others prefer to purchase packages whenever a cost-effective package exists.

Choosing a recommended alternative for the proposed system requires some distinct skills, including (1) an understanding of strategy alignment to identify the important values for the new system; (2) innovation or creativity to identify feature and sourcing options that support organizational values; and (3) a framework within which to evaluate the alternatives and choose a recommended solution for system delivery. The two essential requirements for a recommended solution are (1) the client accepts the solution and (2) the solution positively impacts the performance measures for objectives identified during the strategic alignment.

ALTERNATIVE SOLUTIONS

When a team can think of only one way to solve a problem, then the team may not really understand the problem and the solution may work poorly. While focusing on a plan of action as early as possible is convenient, this approach often leads to incremental modifications or to radical leaps of faith. Part of a prudent early analysis is to formulate several realistic feature and sourcing **alternatives** and then compare them. Sometimes the alternatives create useful new revelations, and sometimes they confirm that the current system is not badly broken. In most situations, the team should be able to come up with three or more reasonable alternatives.

Choosing a Design Option

Design options represent the overall design approach that the team selects to obtain the proposed system. Design alternatives include the following:

- *Use the current system.* With minor modifications, the current operation may provide the desired functions and performance at low risk and cost and sooner than most other alternatives. Correcting the obvious flaws in an existing system may cost little and may offer a good alternative to the time, risk, and expense for developing a new system. Clients may become disenchanted with a system and may want to discard it although a small effort to fix the problems or to provide appropriate training on its use may convert it into a desirable and cost-effective alternative.
- *Modify a current system.* This option focuses on a current system as a starting point for synthesis of an alternative. Many projects use some variation of a current system to create proposed system alternatives. The team may change the features and/or sourcing or may replace the entire current system with an updated version. A common variation involves taking a current system from another organization, perhaps a competitor or one in a different business, and modifying it for use in the client’s organization.

Changing to a different physical infrastructure of software and hardware products represents a common modification to a current system that often

improves performance. The most significant modification in the GB Video case involves converting to a computer-based infrastructure. Many projects in the last few years have involved converting applications from mainframes to server networks or Web-based systems often with little change in functionality. Another common form of modification adds new functionality with or without changes in infrastructure. For example, a GB Video Web-based system might add a video availability function that allows customers to view the availability of videos from a Web site display, a function that lies outside the current system.

- *Apply zero-base design.* Zero-base design options discard or ignore the current operation and focus on the strategic framework as a starting point to synthesize an alternative. For the area under study, the team searches for the feature and sourcing choices that contribute to the strategic goals, objectives, and performance measures of the organization. When no current operation or system exists, zero-base design offers the only option.

For each design option, the team may choose from among a number of function, performance, sourcing, and infrastructure options. For example, the team may hire an application service provider (ASP) to make the minor fixes and run the current system or to modify and/or run the current system. Or the team may replace the current system with a purchased package system. The package may have more, the same, or less functionality and/or performance. Many companies contract with an IT design firm or purchase a package system for zero-base design; they do this to acquire quite different functionality and infrastructure than exist in the current operation. With combinations of different design, function, performance, infrastructure, and sourcing options, a team easily can create many different alternatives.

Choosing Functionality

Normally, the client provides the initial set of functions for a proposed system during project definition. During the proposed system specification phase, the team defines and documents the conceptual specifications for functionality in more detail. With a typical conceptual model, a single fixed set of requirements define functionality. During evaluation, the team views functionality differently in two respects. First the team adds the logical and physical dimensions; the team defines the physical and organizational infrastructure and examines the expected performance. Second, functionality becomes a variable; the team considers trade-offs.

With many smaller projects, the team and client easily can identify a reasonable set of functions to include in the system, and further analysis may add little if any value. As projects become large and complex, looking at alternatives with differing levels of functionality becomes more interesting. For these projects, the team, working closely with the client, can try to answer the following questions:

- Can the team find functions, not identified initially by the client, to add to the proposed system? Only functions that result in significant value, that is, have a significant impact on the measures for the objectives identified during strategic analysis, represent good “add-on” candidates.

- Can the team eliminate any of the functions in the conceptual model of the proposed system? Only functions that add significantly to costs with little impact on benefits represent good candidates for elimination. If the team and client did not identify each function as either mandatory or optional while preparing the conceptual model of the proposed system, the team may wish to classify each function at this time. Optional functions represent good candidates for possible elimination in an alternative.

In the GB Video case, an “availability function” might provide a good “add-on” candidate for an alternative. With the availability function, a client might telephone a clerk to determine if the tape or DVD that he or she wants to rent is available, or use the Web site to browse to see what is available. If the customer finds a video that he or she wants, the customer enters a rental via telephone or the Web and arranges for delivery or picks the video up at his or her convenience. Adding this function allows the customer to avoid frustrating trips to the video store when no desirable videos are available. As noted, the availability function represents a good add-on candidate only if it improves such performance measures as rental revenue and customer loyalty.

In the GB Video case, the team might consider an alternative of eliminating the functional requirement to go through the member process for every rental. With the new alternative, the customer could access the rental function directly. This change in functionality represents a good candidate only if it saves costs or provides benefits. For example, perhaps customer satisfaction will increase if the customer doesn’t have to wait for the system to go through the member process. Or perhaps most package systems do not include this feature but handle membership-checking in another way.

Choosing a Sourcing Option

Once a team decides what a new system should do, the team can then decide how best to obtain the solution. In the early years of computing, most organizations found only one sourcing option: build. The option to purchase package systems did not exist. Today, organizations can choose from a wide range of **sourcing options**—to build in-house, to lease a package, to buy a package, to contract with a vendor for a custom package, or to hire an application service provider to take over operation of the entire function.

Many organizations look first at the purchase of a package—typically the low-cost, low-risk alternative. If a suitable package exists, the use of the package often becomes the recommended alternative. When no suitable package exists, then the organization may consider in-house development, contract development, or ASP alternatives. For very small organizations, some form of outsourcing may offer the only feasible alternative for a solution. Larger companies may use some form of outsourcing for noncore applications and reserve internal development efforts for systems that involve proprietary data or provide competitive advantage.

Most teams and clients feel comfortable with selecting sourcing options. Many clients bring prior experience with making sourcing decisions in other areas of

the organization. Most people make personal sourcing decisions every day, for example, buy a house or rent an apartment, paint your house or hire a painter, fix a meal or eat out, select the specific restaurant when you do decide to eat out. While often summarized as “build or buy,” information system sourcing actually involves a very wide range of alternatives.

Building a Solution In-house

Clients may decide to build a solution in-house. This option means the organization takes the primary responsibility for building the application using its own staff members. Consultants or third-party staff often may perform part of the work on the project. Some of the situations that lead to building an application in-house include:

- Modify a legacy system or create a new system to work with a legacy system. In-house people already know the legacy system while outsiders will face a more costly and time-consuming learning curve.
- Modify or create a mission-critical system. The organization does not want outsiders to gain knowledge of the proprietary features or competitive information associated with the system.
- Build a small application where the administrative costs of outsourcing may exceed the cost to build the application.
- Build “institution-specific” applications. The organization may believe that they can build certain applications faster and for less cost than any outsourcer. For example, organizations often decide to build customized decision support or information presentation systems to fit the preferences of management. With institutional knowledge and modern tools, the in-house staff quickly can generate custom access to data tailored to the preferences of the managers. Tailoring a purchased package to present the desired data and interfacing the package with the existing transaction systems often costs more. For similar reasons, organizations often build simple Web display programs.

Outsourcing subalternatives related to building a system in-house include hiring contract people and consultants to assist the in-house staff and outsourcing programming. Using contract people to cover workload peaks often incurs less expense than expanding the in-house staff. Consultants bring skills and knowledge that may not exist in-house and may cost a lot to develop in-house. Some organizations hire an outside project manager and use primarily their own staff to do the work. Once complete detailed specifications exist, many domestic and international vendors perform programming at a low unit cost. Much of the programming work at a large number of U.S. organizations today is outsourced, often to overseas companies.

Outsourcing the Solution

Solution outsourcing occurs when the client gives primary responsibility for an application to an external party. Representatives of the client may work with the

vendor to monitor progress and quality and to make decisions as needed. Staff members of the client may perform part of the work. Some of the situations that may lead to solution outsourcing include:

- Small organizations may not possess the information technology (IT) people needed to build and maintain a system in-house.
- When a number of suitable package programs exist for an application, the costs and risks of building a system in-house often exceed those for purchasing the package.
- Organizations may outsource some or all of the less mission-critical systems in order to focus in-house IS people on higher priority projects and/or to smooth out the workload.
- Multiuser transaction processing systems (such as airline reservation services, catalog sales, and such) are good candidates for outsourcing. Building transaction systems with good performance, good audit trails, backup and bulletproof interfaces takes major time and specialized skills. If a team can find an appropriate package system, the package may represent the cost-effective solution.
- Systems that require skills and experience that the organization does not possess are good candidates for outsourcing. Outsourcing may provide the desirable alternative for projects or parts of projects that involve such issues as new languages, new hardware, complex logic, high levels of integration, and special security requirements. Working in new environments can require high training and learning curve costs. Designing a system in a new environment involves a high risk of failure and/or serious cost and time overruns.

Writing good application software poses a difficult, time-consuming task. Large computer systems are some of the most complex systems ever created. Major applications can contain hundreds of thousands of code statements. Each code statement is a potential source of error. Creating clean tested code takes a long time. Some estimates of programmer efficiency suggest that 25 statements of debugged, tested code per day represent a reasonable output for an experienced person. Most analysts can develop a system that satisfies a few basic functional requirements. However, adding the edit checks, audit trails, optional dialogs, and exception processing to create an effective operational system rapidly becomes complex.

As noted earlier, the client may find it difficult to specify the logic and other specifications needed to solve the problem that the client faces. Particularly in smaller organizations, a client may know the desired outcome but know little about how the system should achieve the desired outcome. Purchased systems incorporate a set of solutions and options that have satisfied the needs of a number of previous customers. The package system's developers have researched, implemented, revised, optimized, tested, and packaged appropriate functions to form a complete system.

Early purchased systems offered a limited set of options. If the vendor would provide the source code, customers who disliked the design decisions made by the vendor sometimes revised the logic of the purchased package by changing the code. These revisions could cause serious problems. When a problem arises

in the modified system, the package vendor blames the customer's modification and the customer blames the purchased system. A further disadvantage of customer-modified code occurs when a vendor releases new versions of the package to correct errors and add features. Vendors may put out several new releases a year and often stop providing support for the older releases. A customer with a modified package faces difficult choices—either stay with the old release and give up the improvements and vendor support or incur the modification cost and hassles all over again for each new release.

Package and service developers have improved their products by including a number of customization options. Some types of customization are quick and relatively easy. For example, Microsoft Office products work without customization, or the user, if desired, can activate or deactivate a number of functions and select from many different displays quickly and with little if any training. At the other extreme, the SAP Enterprise Resource Planning System usually takes a substantial team of analysts several years to customize the system for the client organization's requirements.

Even with customization options, clients often want to modify packages. The more thoughtful IT managers have learned from experience that the only reasonable way to customize a package is to leave the package code intact and “wrap” the package in customized interface programs. Packages often contain application program interfaces (APIs) to facilitate the wrap process. For example, a customer who buys a package without a Web option may write Web-enabled screens that capture, edit, format, and submit data to the package through an interface. Customization beyond wraps that change the package code can cause serious problems as noted earlier. Organizations contemplating the purchase of a package system should resolve to live with the code in the package even if the purchaser must modify existing rules and procedures to accommodate the new system. A more complete discussion of the issues associated with outsourcing appears in Chapter 10. A team that plans to recommend an outsourcing option may wish to read Chapter 10 before proceeding.

Many vendors offer **package systems** for such common organizational functions as budgeting, accounting, purchasing, payroll, personnel, project management, and hundreds of others. Some packages focus on small organizations while others handle the largest multinational corporations. Several vendors, for example, SAP, Oracle, and PeopleSoft, provide enterprise resource planning (ERP) packages, which are a group of integrated systems that cover a broad set of core functions. Packages represent a form of resource sharing, that is, the multiple users of the package can share the cost of development and modification. Advantages of package systems may include:

- The package may cost less and/or provide an operational system much faster and/or involve lower risks than building in-house.
- Generally, lower levels of skills and experience are required to implement a package than to build one.
- The package vendor may agree to maintain and update the system either at no cost or for a fee.

- Many packages come with such features as I/O interfaces, logic, error checking, and performance, that the vendor has field-tested in a number of prior implementations.

Possible disadvantages of packages include:

- The time and cost of implementing a package in an organization may amount to far more than the client expects.
- Packages provide only a predefined set of functions. While some packages contain many options, they may not contain all the functions the client wants. Modifying a package to meet client desires negates many of the advantages.
- The package may not provide all of the functionality and performance claimed by the vendor.
- The package may become an orphan when the vendor goes out of business and/or stops supporting the package.

With **contract development**, consulting companies and other vendors, for example, Accenture and IBM Global Services, build a system to the client's specifications. The contractor may assume full responsibility for building and delivering a solution and/or may work in a partnership with the client. For some projects, the contractor works on a fixed price; for others, the contractor receives actual costs incurred plus a fee. Advantages of contract development may include:

- The contractor may possess the skills and experience that are lacking on the client's staff.
- The contractor may have built similar applications for other clients.
- The contractor frees up the client's staff for other projects.

Possible disadvantages of contract development include:

- The resulting product may not arrive on time, may not work at all or as specified, and/or may cost much more than expected.
- When the client has little involvement in building the product, the client may experience major problems maintaining the system.
- Contract development may cost more than building in-house or using a package.

A number of **application service providers (ASP)** offer full-service outsourcing. An ASP outsourcer accepts the primary responsibility for running an application including entering the inputs, doing the processing, maintaining the files, and preparing the outputs. The ASP may develop and maintain the application program or use one supplied by the client or a third party. Often, the ASP provides the facility and hardware to run the application. Advantages of ASPs may include:

- The client need not commit any resources except money to the project. The ASP may provide facilities, equipment, operating staff, and maintenance.
- An ASP with a number of customers can provide frequent updates for volatile situations at a reasonable price, because the costs are shared by all the customers.

Possible disadvantages of ASPs include:

- The ASP and the client may operate with conflicting goals. The ASP wants to run the application at the lowest possible cost to increase the ASP's profit while the client wants the best possible results and service.
- The ASP may go out of business or discontinue the service leaving the customers in a difficult situation.
- The ASP may respond slowly, ineffectively, or not at all to user and customer concerns.

Choosing Infrastructure

Selecting physical infrastructure often is either very complicated or very easy. For small projects in large organizations, the team often must use either the physical infrastructure that exists or select from a standard list, for example, the organization's IT group may require the use of specific servers running a specific operating system with a specific database engine. For many of the smaller projects typical to class field projects, almost any server and database engine will execute the desired functions. For large projects, selecting the appropriate physical infrastructure requires a lot of skill and experience plus some good luck. Many larger organizations hire staff members who specialize in physical infrastructure design.

Since many infrastructures will execute the desired functions, the major problems with infrastructure selection are achieving interoperability and the desired performance at a reasonable cost. **Interoperability** is the ability of different software and hardware components to work together correctly and effectively. Only certain combinations of servers, operating system, database engines, and other components work well with each other. Many projects also involve specialized hardware—for example, communication networks, bar-code scanners, optical data readers, and point-of-sale devices that introduce additional interoperability issues. Performance evaluation is discussed in the next section.

When the team must select or recommend infrastructure, some guidelines are as follows:

- Consult with the client's infrastructure expert if one exists.
- Talk with people from similar organizations to find out what infrastructure they use.
- If the client has a major or dominant infrastructure vendor, ask the vendor representative for suggestions. The team also can look on the Web or contact vendors to find out what is available.
- Give preference to infrastructure from established vendors, that is, vendors with a multiyear record of supplying products to a number of satisfied customers.
- Give preference to infrastructure products that fit the existing skills and experience of the client's IT staff members.
- Give preference to "expandable" infrastructure products. For example, design the processing infrastructure so that the client can add memory and additional processors to a server or can add more servers if more capacity is required.

Evaluating Performance

Complex interactions between the physical infrastructure, system functionality, user behavior, load, file or database sizes, program design, and other factors determine **performance**. While cost-effective program execution or run times remain a consideration, most performance issues focus on **response times** for interactive systems, system availability, and load capacities. Response time is the interval between the time a user makes a request and the time that the user receives a response. For GB Video, the response time is how long it takes from the time that the user enters a member number and clicks on the “Find customer record” button to the time that the customer data appears. A second or two may seem immediate, 10 seconds or more may become annoying, and a minute or more probably becomes almost unbearable.

Most small single-user systems tend to provide good response times given the speed of modern hardware. Response times become an issue mostly in multi-user systems with large files and complex logic. A system may produce good response with only a couple of users but slow way down as the number of online users increases. In similar fashion, large file sizes and complex multitable queries may slow down response time in multiuser systems.

Availability is the percent of time the system is operating correctly during the interval the users want to access the system. Both hardware failures and program errors may reduce availability. A number of equipment vendors offer hardware with features to enhance availability. When a system supports an online revenue generating function—for example, airline reservations, or catalog sales—most clients want availability well over 99 percent. With an informational Web site or an off-line application, a client may accept lower availability. GB Video probably wants over 99 percent availability for the Rental and Return system, but accounting can accept lower availability for their system to produce reports.

Estimating performance presents major difficulties. Most prototypes and demonstration packages show very good response times because they operate with a single user and small file sizes. Prototypes and demo packages tend to provide little or no insight on the response time or availability in the actual system. Most student project teams possess little ability to evaluate system performance. Some client organizations may contain people who can make reasonable estimates of performance. For large online systems, especially package systems, many clients want to see a full-scale performance test. The organization will run the actual package on the physical infrastructure the client plans to use with a copy of the actual client database and an appropriate number of real or simulated users.

In the modern IT world, load-related performance problems tend to cause less havoc than they did in the legacy mainframe world. Since many modern systems use off-the-shelf hardware, clients can quickly and inexpensively add additional hardware to improve poor response times or to increase capacity, especially if the infrastructure was designed to be “expandable.”

Describing Alternative Solutions

For an alternative to be useful, the team must describe it in sufficient detail for both team and client to understand how it will work—features, response time, capacity, and availability; what actions and costs are required to implement it;

and what benefits will result. Many alternatives sound great when stated in the one paragraph but fall apart on a more specific examination of the functionality, performance, required infrastructure, and organization. A comprehensive description of each alternative forces the team to think through the issues, allows the clients to decide if the system contains the appropriate functionality, and gives the evaluation base for a reasonable comparison of alternatives.

The description of the current system or situation provides a baseline. The descriptions of the other alternatives can avoid a large amount of redundancy by focusing on only those features that are major differences from the current situation. A full description for each alternative contains three parts:

1. A clear and explicit title that describes the system. The title "Option 1" only makes sense to the design team while most clients will understand the title "Current System."
2. A comprehensive description. A good description starts with a one-paragraph summary of the alternative. Additional paragraphs can provide a general description of these critical features:
 - a. *Data structure.* Significant changes from the current system, including new entities, additional attributes, and new or changed relationships.
 - b. *Logic and functionality.* Features that are added, deleted, or changed. A high-level DFD can illustrate the differences graphically.
 - c. *Architecture.* Specifications for additional or modified hardware, software, and more are needed for each alternative. An alternative that introduces a completely new environment or adds some new components into an existing infrastructure raises concerns about the interoperability between such components as networks, application packages, DBMS, transaction monitors, operating systems, and processors. A diagram displaying the new architecture can help when significant changes are proposed.
3. An **evaluation**, or an estimate of the merit of a solution. Normally the discussion of each alternative concludes with an evaluation of the alternative. The evaluation may include:
 - a. An analysis of risk and feasibility.
 - b. A listing of advantages and disadvantages.
 - c. An economic or cost/benefit analysis.

An example with descriptions for the GB Video alternative solutions appears in Figure 9.2 at the end of the next section on evaluation.

EVALUATION

Evaluating alternatives for their impact on organizational performance forms a critical step in information system design and one that often is forgotten in the rush to build or buy something. The widespread use of rapid development approaches for IT design makes the performance-oriented design framework even more critical. Some teams and managers focus from the beginning on only one alternative, omit any evaluation and go directly to detail design or procurement.

At a minimum the team should spend one or several hours evaluating alternatives against the strategic framework for the project. Without at least a minimal, performance-oriented evaluation, the team has no reasonable way to choose a satisfactory set of function, infrastructure, and procurement options. For large, expensive projects, much more evaluation effort is warranted.

The organization's strategic and tactical requirements provide the basis for evaluation and choice of alternatives. Sponsors, clients, senior managers, users, customers, and other non-IS people make the judgments about the value that an alternative adds to an organization. These value judges care little or nothing about the elegance of the design, the programming languages used, or any other IT issues. Their judgments use the answers to such questions as:

- Does the alternative help the organization compete or fulfill its mission?
- Will the alternative contribute to organizational performance? Will it increase sales and profits, reduce costs, improve response time, or improve quality?

In short, the value judges ask the questions that the team studied in the strategic alignment part of project definition.

As noted in Chapter 6, the strategic alignment for a project involves vision, mission, strategies, objectives, and performance measures. The team uses the strategic framework to examine each alternative to determine its contribution to performance. Once the performance contributions are identified, the team should "test" the accuracy and realism of the results. Every alternative deserves at least a proforma test; try out some scenarios and work up the numbers to show how the system actually might achieve the performance improvements. Many projects that sound good during project definition require unrealistic assumptions, for example, a 200 percent increase in sales in a mature market to produce the claimed benefits.

Choosing the alternative to recommend represents one of the more complex and difficult tasks the team will face. In micro economic theory, the evaluation and choice process is simple: from a universe of all possible alternatives, select the alternative that maximizes (total benefits – total costs). Unfortunately, the real world poses many difficulties. The team knows only a few of the alternatives, not all the possible ones. Even worse, estimates of total cost and benefits made prior to detail design and implementation prove notoriously unreliable. Costs and benefits arise over time leading to time-value-of-money issues. Most alternatives contain substantial risk. Will the new system work? Can the team complete it on schedule? How long will the organization use the new system? The endless rapid emergence of new technology adds more uncertainty and further complicates evaluation because new technologies enable new alternatives about which little or nothing is known at the time of the evaluation. The life of the proposed system also is uncertain; the system may stay in service for many years or never become operational. With high levels of uncertainty, risk becomes an important, if not the most important, part of selecting the recommended solution.

Because of the difficulties of evaluation, some teams give up after a brief look at the evaluation process and hope for the best. The better teams, however, use their best efforts and recognize that many sources of error exist. For example, the team may use formal evaluation methods to eliminate clearly poor choices.

If several “similar” and acceptable choices remain, the team and client can select the recommended solution using client preferences and/or organizational policy and culture. As noted earlier, the team does not have to find the best alternative—finding a reasonably good or satisfactory one often meets the needs of the organization. March and Simon (1958) call this concept “satisficing” and suggest that satisficing is the most feasible way to select most solutions.

Reasonable guidelines for a practical approach to evaluation suggest that the team begin by conducting a **feasibility analysis**, a broad, high-level evaluation, to separate alternatives into more promising and less promising groups. Often, a very limited evaluation will show that an alternative does not meet the constraints of the client or is less desirable than other alternatives. These less promising alternatives warrant little if any further evaluation. For example, if building a system in-house costs several times as much as buying exactly the same functionality and performance in a package system, the team and client can drop further analysis of the build option.

Feasibility analysis also can help the team to avoid further consideration of “likely disasters”—solutions that contain a much larger risk of a serious negative impact on profits and/or critical performance measures of other alternatives with similar benefits. A single large, long, high-cost project contains more disaster potential than several phased smaller, shorter, less expensive ones that accomplish the same result. When an unsuccessful project can lead to disaster (bankruptcy, loss of key markets, etc.) the team should try to rule out such high-risk alternatives as outsourcing to unknown or unproven vendors, buying package systems that lack adequate backup options, building alternatives that require skills and resources the organization does not currently possess, or selecting alternatives that require unreasonably short deadlines.

Only the more promising alternatives are candidates for in-depth evaluation by the team. For the more promising alternatives, the team can consider such evaluation methods as these:

- **Cost/Benefit analysis.** Identify and quantify the costs and benefits for a solution and calculate a measure of merit, for example, net present value (NPV), return on investment (ROI), or payback period. Select the alternative that scores highest on the selected measure. Organizations often insist on cost/benefit analysis for projects that involve a large front-end investment of capital.
- **Features analysis.** Identify the features that are most relevant to the selection of a recommended solution and assign a measure of merit (yes/no or numerical value) for each of the features in each of the alternative solutions. Organizations often use features analysis for smaller projects and sometimes for larger ones when cost/benefit analysis appears unfeasible.

Feasibility and in-depth evaluation approaches are discussed in more detail in the following sections.

Feasibility

In a feasibility analysis, a broad evaluation overview, the team asks if a solution can meet the schedule, cost, technical, legal, security, organizational, outcome,

and risk constraints of the client's organization. The full evaluation of a solution alternative can involve large amounts of time and effort. The team wants to devote this time and effort only to alternatives that meet the client's broad constraints. A feasibility review may allow the team to eliminate infeasible alternatives. The team normally conducts a feasibility review early in the project during project definition and revisits the question of feasibility at every major decision point in the project. When the team is unable to find any satisfactory alternative to meet the client's constraints, the team needs to review constraints and goals with the client.

Schedule and cost feasibility refer to whether the team and client can implement the system within the time and cost constraints determined by the client. Clients often have a limit on the money they are willing to expend for a system. If a client sets a limit of \$200,000, then the team can consider eliminating alternatives that cost substantially more. The team also can consider eliminating alternatives that fail to meet the client's schedule. For example, if GB Video wishes to begin catalog sales on May 15, then the team and client may reject a "low-cost" vendor who proposes a June 15 date because of schedule infeasibility. Sometimes clients choose to start with a bare-bones package or an evolutionary prototype solution to get the essential features operating within a schedule constraint.

Technical feasibility focuses on the ability of (1) available technology to handle the system functions and performance and (2) the available staff to use the technology. The analysis focuses on the "technological stretch" necessary to develop the solution. A solution that builds on technology already in use at the client's organization requires little or no stretch. A solution that uses complex technologies never before used by the client and with which the client's staff have no experience requires a huge "stretch." Such a solution may be technologically infeasible. A solution that uses complex technologies for the first time ever anywhere runs a very large risk of technological infeasibility.

Legal feasibility examines the legal and ethical issues associated with the project. Privacy and licensing issues are common sources of legal and ethical problems. For example, a loan application system that uses information on the applicant's race or religion probably is legally infeasible. In many areas, for example, accounting and finance systems, human resource management systems, automobile manufacturing, airline operations, food processing, and pharmaceuticals, a large number of government regulations apply. Systems that do not meet or accommodate these regulations probably are legally infeasible. A system that uses customer information for purposes unrelated to the reason the customer supplied the information, even if legal, may raise questions of ethical feasibility.

Operational feasibility assesses the impact of the solution on the way the organization operates. When solutions require major reorganization or major changes in existing procedures, the people in the organization may be unable or unwilling to operate within the changed structure. Package systems often introduce issues of operational feasibility because packages assume an organizational model that may not match the existing one. With a package system solution, the client must decide if the people in the organization are willing and able to make the operational and policy changes required to utilize the package. A system built

in-house to automate existing processes often has little impact on the organization and thus raises few questions of organizational feasibility.

Outcome feasibility refers to whether or not the alternative can achieve the outcome desired by the client. For example, when a client has a major goal of increasing sales, a feasible solution enhances the competitive advantage or position of the client. An infeasible solution may have little or no impact on sales or worse, may introduce new competitors, or may restrict the client's suppliers or customers in a way that limits sales growth.

Security feasibility represents a major issue for many projects. A package that offers many features and good benefits may end up as infeasible because of a lack of adequate security. Financial, military, and other organizations may rank security as the first consideration. Web access generally raises security issues and may necessitate a firewall or even a separate network. Data stores raise both access control and protection, or backup, issues. Because security issues involve complexity and continuous change, organizations often hire consultants to evaluate and specify appropriate security measures.

The overriding considerations for a solution may influence or determine feasibility. Overriding considerations mandate or strongly direct the team toward or away from a specific solution and can include:

- *Competitive necessity.* The organization must implement an alternative or one of a class of alternative solutions to avoid such severe consequences as loss of market share or profits. Several bookstore chains implemented Web-based catalog sale systems to protect their market from a Web-based competitor.
- *Client directive.* A client with authority and/or funds wants a specific alternative. The team should perform an evaluation of alternatives and share the evaluation with the client. However, if the client continues to demand a specific solution, the team normally will proceed to system delivery with the client's recommended solution.

Risk Analysis

The level of **risk** associated with a project often represents an evaluation measure of high importance to the client and, as noted earlier, high risk solutions may fail the feasibility test. Risk means that a project may realize less desirable outcomes, such as extended completion date, higher costs, lower benefits, and less functionality, than forecast or planned. All projects involve risk that can arise from one or more of the following sources:

- **Technology and vendor.** A successful IT project requires technology products delivered on time that work as expected. Even with the best of intentions, vendors and customers often miscommunicate. The vendor may not deliver the product at the time the client expects. When delivered, the products may not work as expected. For example, package program vendors sometimes tell customers that the desired features are in the next release due in six months. The features actually may appear in six months, in several years, or never. Or the vendor literature may say that a package works with a particular operating

system when it actually works only with certain releases of the operating system or only with the operating system running on certain servers.

- **Staff.** A successful project requires people with time to spend on the project, who have skills, experience, and dedication. A project may lack the necessary people from the beginning or may start with adequate people and then lose them. For example, the existing IT and/or functional staff members may lack the necessary knowledge or experience and/or the organization may not hire, train, and assign the people required for the project. During the project, the organization may assign the needed people to other projects, or the key staff members may leave for other jobs. The IT organization probably has more control over staff risk than over risk from other sources.
- **Sponsor.** A successful project requires a sponsor to fund and defend the project. The key sponsor may leave, change his/her mind about the project, be overruled by a superior, get into a control struggle with another senior person, demand features that are impossible to implement or do not work, set impossible deadlines or budgets, run out of money, or for some other reason withdraw support for the project.
- **Competition.** The value of a system often depends on the competitive environment. For example, a system may promise large benefits because it offers features that the competitors do not have. However, by the time the system is ready to use, the competitors may have as good or better systems in place. Changes in the competitive environment can bring disastrous consequences to IT projects.

In concept, risk is expressed by a probability distribution of outcomes. For example, a project that the team estimates may cost \$1.5 million might have the following distribution of cost outcomes:

- Cost = \$1 million with probability = .1
- Cost = \$1.5 million with probability = .7
- Cost = \$10 million with probability = .2

The team could use the \$1.5 million cost estimate in a cost/benefit analysis and say no more. But the solution might cost as much as \$10 million. If the system is mission critical and the company cannot afford \$10 million, this alternative might lead to a “disaster.” The client may wish to reject the solution because of the disaster potential.

In practice, the team seldom can estimate the distribution of possible outcomes. However, the team can help the client to reduce risk. Possible actions include those that follow:

- *Follow safe design practices.* For example, limit project scope, strive for short development times, use non-mission-critical applications to experiment with new technologies, purchase infrastructure components from the industry leaders, hire a consultant or contractor if the internal staff does not have the appropriate skills and experience, and build some slack into the budget and the schedule.

- *Practice safe communication.* Keep the client informed of possible risks and tell the client immediately and repeatedly when problems occur.
- *If a solution with a substantial risk of a disaster appears to offer the best or only viable choice, make absolutely certain that both the client and sponsor understand the risks.* In some cases, a solution with a substantial risk of disaster may offer the only viable choice, but the client and sponsor need to fully understand the situation.
- *If at all possible, provide backup for a high risk or possible disaster alternative.* For example, a catalog company project to develop its first Web solution for the order-taking process probably represents a high-risk project. To protect itself, the company might add capacity to the existing telephone order system and maintain the telephone system to work effectively until the new solution passes all the tests and begins operation.
- *Use a shorter payback period criterion or a higher internal rate of return for the higher risk alternatives.*
- *Use intuition and judgment to get a ranking of alternatives that combines cost/benefit and risk.*

Deriving estimates for risk can present difficulties. The more experience the analysts and clients have with IT development, the more likely they are to make reasonable judgments about risk. Some helpful checkpoints are listed below:

- *Risk goes up with size as measured in person months to complete the project.* Small projects, for example, projects that require a couple of person months, are low risk, because they are easy to manage and if they fail, not much is lost. For the same reasons, large projects, which require tens of person years, are higher risk. Cost risk is relative to the size of the organization. A \$1 million project may present low risk for an organization with \$1 billion of annual revenues but very high risk for an organization with only several million dollars of annual revenues.
- *Risk goes up with lack of familiarity and experience.* An experienced mainframe COBOL programmer can develop a new COBOL application similar to existing ones with low risk. The same programmer developing a first application in Visual Basic and Oracle to run on servers in a distributed network faces a much higher risk. Everything in the development environment has changed.
- *Risk goes up with emerging technologies.* Employing leading-edge technologies introduces not only the lack of familiarity with the technology but also the risk of unexpected problems with the technology itself. For example, the technology may lose its market share and disappear.

In virtually every project and for every alternative solution, a careful evaluation of risk provides a valuable input to the choice of the recommended solution.

Cost/Benefit Analysis

Applying cost/benefit analysis for a set of alternatives consists of three steps: (1) identify the benefits and the costs for each alternative; (2) select an evaluation metric—net present value, payback period, and so on, and (3) rank the alternatives.

Without careful thought and a plan, the team may focus on costs, which are the “easiest” part to measure, and neglect benefits. If the performance-oriented design phase was conducted correctly, many of the benefits of the applications already are identified, for example, revenue enhancement, cost reduction, customer service, employee productivity, or quality improvements. A good way to get a desirable cost/benefit performance from a new system is to (1) set specifications to achieve the desired benefits and to (2) use costs for guidance on the most effective way to achieve benefits.

Costs and benefits may occur as one-time or ongoing amounts. One-time amounts are usually associated with solution delivery—development costs, hardware and package purchases, initial training, and so on. Ongoing amounts recur on a monthly or annual basis—operating costs, yearly fees, and such.

Benefits

Estimating benefits raises a number of challenging problems. The major benefit from a computer system often comes from changing the way the organization operates. As a result, the team must estimate benefits for a situation that currently does not exist. If the IT team members possess expertise only in IT, the team members may not understand the impact of their system on the organization’s performance measures. For example, the IT team might consider the potential for enrollment growth as a major benefit of a new enrollment system, while the university administration wants to keep enrollment constant. A good strategic alignment analysis (see Chapter 6) can help the team to avoid this problem.

The team probably will find neither the time nor effort to evaluate every potential benefit in depth. In this event the team can apply the 80/20 rule: focus on the 20 percent of benefits that give the most return and make fast, rough estimates for the others. In evaluating ongoing benefits, the team should recognize that any competitive advantage from an IT solution might experience a short life. For example, the advantage gained by the first firm to use a targeted merchandising system may disappear in a few years as other firms catch up. A well-designed information system offers a vehicle for continuing innovation, not a finished product.

Information technologies substitute capital for labor, a traditional and effective way to achieve higher productivity and lower costs. To use information technology, the designer invests in hardware, DBMS, application packages, tools, other software, development, training, and implementation costs. The benefits can range from such local measures as reductions in programmer time and clerical time, to such broader issues as increased engineering, manufacturing, or marketing productivity. Benefits from an IT solution can arise in the following ways:

- **Cost reduction.** Some applications, for example, payroll, provide a more or less standard functionality. The benefits come from achieving the standard functionality at the lowest possible cost. A number of proposed systems may perform the same functions as the current system but offer cost savings as the benefit. Information systems also can reduce costs by substituting for other

higher cost resource inputs. For example, electronic data interchange in firms and between firms combined with new systems may allow lower inventories.

- **Cost displacement.** A form of cost reduction, cost displacement substitutes a new lower cost process for a more expensive current process. A system that allows GB customers to enter video orders themselves on a terminal shows an example of a cost displacement system. The system replaces paid clerk time with unpaid customer time. Many retail stores now provide self-checkout for customers—a cost displacement or disintermediation option marketed to customers as a “time saver.”
- **Cost avoidance.** Some solutions provide the benefit of avoiding costs that otherwise might occur in the future. Having a new solution may allow the firm to avoid such costs as ones arising from the loss of business to competitors or payments for penalties and fines.
- **Revenue and performance enhancement.** Increasingly IT creates products, services, and situations that enhance revenue or another important measure of performance identified during strategic alignment. For example, Dell’s automated manufacturing control system produces quality levels in PC and server products that appeared unattainable a few years ago and allow Dell to steadily increase market share and revenue. Wal-Mart’s merchandising systems allow each store in the world’s largest retail chain to increase revenues by serving the needs of the customers in its specific area. The performance measures for some strategic objectives involve metrics not directly translatable to service, profits, or revenues, such as quality and customer satisfaction.
- **Risk reduction.** Many organizations are “risk adverse,” in that they value reductions in the risk of incurring large losses. Sometimes risk reduction comes from conservative compliance with laws or regulations. In other cases the risk reduction may come from a system with low initial investment and/or low shut-down costs. In the event the project fails, the organization will survive.

Intangible benefits no doubt exist but are subject to abuse. Unsupported and unmeasured claims of wonderful intangible benefits, if accepted at face value, can justify even the worst of alternatives. For example, a cost/benefit analyst might claim that the system will provide benefits of over a million dollars from empowering users and strengthening morale. Empowering users and strengthening morale probably is good, but what performance measures identified during the strategic alignment analysis will change, by how much, and why? As a general rule, the team should (1) to the extent possible, translate the intangible benefits into measurable changes in the key performance indicators and (2) use any remaining, unevaluated intangible benefits to break near ties between alternatives.

Costs

Projects incur costs for a wide variety of goods and services; and, in most projects, costs are easier to identify than benefits. Typically, the wage costs of people, not the capital costs of computer equipment, are the largest cost component. With extensive outsourcing, the outsource contract costs exceed the cost of people

on staff, but the vendor probably is billing mostly for personnel costs. Possible sources or categories of cost include:

- **Personnel**
 - Clients and users
 - Managers
 - Project staff
 - Implementers
 - Trainers
 - Consultants
 - Operators
 - Maintainers
 - Security
- **Facilities**
 - Rental or use charge
 - Construction
 - Operation—utilities, maintenance
 - Security features
 - Furniture and supplies
- **Information technology**
 - Hardware
 - Software
 - Communications use
 - Maintenance and upgrade fees
 - Supplies

Total costs include both **one-time** and **ongoing costs** over the life of the system for the following categories:

- *Initial development.* Many costs occur prior to start-up of the system including team salaries, consultants, hardware, system and application software, network connections, and development tools.
- *Implementation.* A second major group of costs occurs at the time the new system goes into operation including data conversion, testing, training, parallel operation, and changing organization and procedures.
- *Continuing operations.* A final group of costs may continue to accrue over the entire life of the system including utilities, hardware and software, ongoing training, maintenance and upgrades, and people to operate and use the system.

The accumulation of all of the lifetime costs for a system across all of the organizational units affected by the system is called the **total cost of ownership**. Managing the total cost of ownership is a major goal of many modern organizations. In the legacy mainframe era, many of the costs of a system resided in one cost center. With the distributed systems that exist in many organizations

today, the total system costs exist across the organization in many departments or divisions. To the extent that design, training, operations, and maintenance take place in the decentralized organizations, the costs tend to become commingled and difficult to identify. For example, conversion to distributed systems may appear to save money by reducing the cost of the IT group, when actually many of the costs just shift to other units. If information processing responsibilities are added to operating staff, the system should include the cost of user time devoted to the information processing tasks.

The initial development of a new function or technology in an organization will cost much more than subsequent ones—the traditional learning curve effect. However, initial development of new systems is not the largest cost for many organizations. Studies of IT groups suggest that much of their total people cost and effort goes to training and to software maintenance and upgrade requests. With outsourcing, some or most of these costs may come in bills from the outside vendor. Training costs can exceed even careful estimates by a factor of two or more; and significant training loads can continue indefinitely due to software upgrades, program modifications, and new hires. Because of continuing rapid technology evolution, some training probably will utilize vendor courses. IT staff members may require retraining every several years to maintain technical competence. A typical system with extensive users, for example 100 or more, may incur the cost of a group of people as a focal point for help and/or service, commonly called a help desk or information center. Help providers require training to maintain skills for the applications, equipment, and software in use.

Evaluation Metrics

Organizations that make extensive use of projects tend to establish **evaluation metrics**. An evaluation metric consists of a *method* for arriving at a measure of merit for a project and an *acceptance criterion*, a target that an acceptable project should reach or exceed. Financial analysis methods provide a framework for computing a sum of *net benefits*, the benefits for the period minus the costs, for each period, over the life of a project to get a measure of merit. Commonly used methods include net present value (NPV), return on investment (ROI), and payback or break even analysis. Each of these approaches can provide helpful information. IT projects contain a large amount of uncertainty over costs and especially benefits. As noted, estimation involves a number of difficult issues. In addition, the environment in which information system solutions operate may change and negate the value of the solution. In this situation, most organizations want to follow a conservative path.

Payback period or **break-even analysis** offers a simple and often effective metric to rank IT alternatives. The payback approach asks how long a period is needed for the sum of the benefits to equal the sum of the costs of designing, procuring, implementing, operating, and maintaining the new system. To use payback analysis, the team simply adds the net benefits for each time period of the project starting at the beginning of the project and going forward along the time line of the system life cycle. Normally the sum starts out negative because a new project incurs costs for some time until it starts to generate benefits. The

TABLE 9.1
The Value of
\$1 Invested at
10 Percent
Annual
Compound
Interest

Year	1	2	5	10	20
Value	\$1.10	\$1.21	\$1.61	\$2.59	\$6.73

payback or break-even period is reached when the sum of the net benefits equals zero. If a new system costs \$10,000 to create and produces net benefits of \$500 a month, the payback period is 20 months, the sum of the costs and benefits equals zero after 20 months of operation. At this point, the organization has earned back all of the money invested in the project.

To follow a conservative approach with payback period analysis, the team can use a short payback period criterion. Organizations typically like to see payback periods of 6 to 24 months for an IT solution. Clearly, payback period is simple to understand and to use. Payback period also deals well with the shortened time frame of IT activities. All time aspects of IT systems, for example, development time, procurement time for purchases, implementation time, and modification time have decreased from the norms of 10 years ago. However, payback period analysis tells little or nothing about the merit of the system over its life cycle. A system with a short payback may produce no further benefits after the payback period while another system with a longer payback period may produce massive amounts of benefits over the life of the system. Many organizations like and use payback analysis because project sponsors want to recover their investment fast and believe that estimating costs and benefits over an extended period in the dynamic modern world is difficult and often misleading.

Time-value-of-money methods weight each net benefit with time and an interest rate. In Western societies, people accept the notion of earning interest on investments. If a person invests \$1 at i percent per year interest, the value of the investment with compound interest after n years is $(1 + i/100)^n$. Compound interest means that the interest earned each period is added to the amount of the investment and earns interest in all following periods. For interest of 10 percent per year, Table 9.1 shows the value of the investment at the end of a selected number of years.

The converse of this relation is that a dollar of net benefit expended or earned n years from now has a *present value* of $\$1/(1 + i/100)^n$. A net benefit of \$1 with a 10 percent interest rate earned 20 years from now has a present value of $\$1/6.73$ or \$0.15. To calculate the net present value and return on investment for a project, the team uses these time-value-of-money relationships to weight the net benefit for each period of the project. The team also can use time-value-of-money calculations to weight the net benefits in payback period analysis; however, payback analysis normally uses the unweighted net benefit numbers because of the short time span involved.

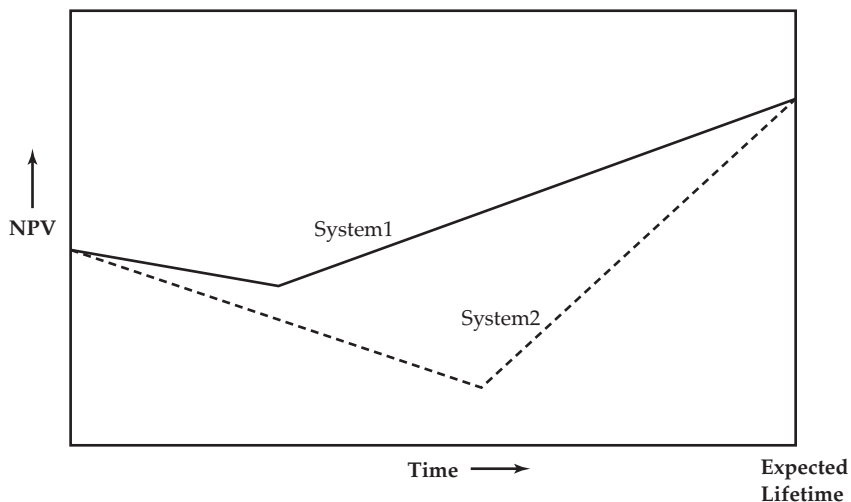
The **net present value (NPV)** for a project is calculated by estimating the net benefits for each period in the life cycle of the project, the same estimates that the team made to calculate payback period. However, the estimates are made over the life of the project. Once the net benefits are estimated for each period,

the team can convert each net benefit to a present value and sum the present values over the life of the project to arrive at the NPV. The client or the organizational standards give the team the interest rate to use, the rate that the organization wants to earn on its investments or selects as its cost of capital. The team can set up a spreadsheet in Excel to calculate the NVP for alternatives or can use one of many package programs for the task.

Many organizations use the NPV method and believe that NPV is helpful or imperative. For example, a system with a short payback may produce a small or even negative NPV, good information to have. However, NPV comes with serious reservations. Since NPV calculates the present value of a stream of future events, NPV requires the team to forecast both the events for each period and how long the system will continue to provide enough value to remain in use. Reality may not follow the estimates. The system may never reach completion. The system may cost more and/or produce fewer benefits than expected if and when it starts to operate. New technologies and changing organizational environments often make an even newer system more desirable than the proposed system long before the end of the forecast life cycle. No one guarantees that IT (or life) is kind or fair.

Two systems with the same NPV can look quite different to a sponsor. The graph in Figure 9.1 shows the NPVs for System1 and System2 over their expected life. System2 costs more initially than System1 and produces higher net benefits once it starts to operate. Although both systems have the same NPV at their respective expected lifetimes, the two probably will look different to a sponsor. A risk-adverse sponsor probably prefers System1 because System1 has a higher NPV than System2 for all the time before the expected end-of-life point. A risk-tolerant sponsor may gamble on getting big net benefits from System2 even past the expected lifetime.

FIGURE 9.1
Two Alternatives with the Same Final NPV



If the team wants to take a conservative approach to an NPV analysis, possible actions include the following:

- Explain the problems of estimating net benefits and life span to the sponsor.
- Set the life span at the point the ability to estimate net benefits begins to decline sharply, somewhere between now and not very long from now.
- Calculate the NPVs for various life spans and prepare a graph with the data.
- Run a sensitivity analysis to show what impact a low and a high estimate of net benefits has on the NPVs.

The **return on investment (ROI)** or **internal rate of return (IRR)** for a project is the interest rate that makes the NPV of the project equal to zero. The team collects the same net benefit information needed for the NPV method and makes the same estimate of the lifetime for the system. The team then finds the interest rate that makes the NPV equal to zero. Teams often use trial and error to find the ROI; try increasingly higher interest rates until one produces a zero NPV. Package programs to compute ROI also exist.

The ROI method provides a way to compare projects of different sizes. One might expect a good large project to produce a higher NPV than a good small project. The size of the project does not affect the ROI. The ROI produces the information that investors often want: the project that produces the highest return on their investments. The ROI method faces all the same problems and limitations as the NPV method. With all evaluations, the team has an obligation to point out that financial measures give an appearance of precision that the estimates from and limitations of the IT project environment may not warrant.

As part of the evaluation metric, many organizations set *acceptance criteria* for IT projects. For example, an acceptable project must have a payback period of less than 18 months or an ROI of 30 percent. Short payback period or high ROI acceptance criteria reflect a conservative approach to recognizing the risks of estimation and of the continuing and rapid change in both IT and the competitive environment.

The Implied Benefits Method

In an ideal world, every project has explicit and measurable benefits. In the real world, the team may find no reasonable way to determine or measure benefits for some projects. One alternative is to look at implied benefits. The **implied benefit** is the benefit amount needed to allow the project to meet the acceptance criteria used by the company. The acceptance criterion can be a payback period or an NPV or ROI over some period, normally the expected life of the system. The implied benefits method will work with any and all acceptance criteria.

For example, a company might decide to create a Web site to display the daily stock prices for the company and a set of its competitors. The sponsor in requesting the project, reasons that the Web site will encourage employees to think about the stock price and about what steps might cause it to increase. A contractor agrees to create the Web site for \$40,000. Yearly operating and maintenance costs will come to \$10,000. For most projects, the company uses an 18-month payback

period as an acceptance criterion, that is, projects that pay back in 18 months or less are accepted. If the stock price application is to pay back in 18 months, then it must produce a monthly *net benefit* (benefits minus operating and maintenance costs) of $40,000/18 = \$2,333$ per month or \$26,000 per year. For the project to meet the company's rules, the implied annual total benefits must be at least \$36,000, the \$26,000 plus the \$10,000 for annual operating and maintenance costs. The team can now ask the client if the Web site is worth \$36,000 a year. Note that the role of the team is to frame meaningful questions; in most cases, the client and sponsor make the value judgments.

Cost/Benefit Table

A table provides a good way to examine the cost/benefit data as illustrated in Table 9.2. The table helps the team to track what and when the costs and benefits occur. The table format also deals directly with the issue of one-time versus recurring costs by simply showing costs in the period that they occur. For short projects of less than a year, the initial project costs often are shown in the first column at time zero, the time at which the system begins to operate. Illustrative tables for the GB Video example appear in Figure 9.2.

Part 1 of Table 9.2 shows for each period during the expected life of the system: the costs, the benefits, the net benefit (benefits – costs for the period), and the cumulative net benefits. The system has an initial development cost of \$10,000 as shown in column zero. In periods 1 to 4, the system operates and incurs costs and generates benefits as shown in the table. The cumulative net benefit is the sum of the net benefit for a period and all previous periods. In this example, a quick glance at the table shows the times at which the cost and benefits occur and that the payback probably occurs at a little over one year if the period of analysis is a year, that is, the cumulative net benefit becomes positive in period 2. A table for an actual alternative could have more detail about costs

TABLE 9.2
A Cost/
Benefit Table

Period	0	1	2	3	4
Part 1. Costs and Benefits for Each Period					
Costs	\$ 10,000	\$ 3,000	\$ 2,000	\$ 2,000	\$ 3,000
Benefits	\$ 0	\$12,000	\$15,000	\$18,000	\$ 9,000
Net benefit (NB)	\$-10,000	\$ 9,000	\$13,000	\$16,000	\$ 6,000
Cumulative NB	\$-10,000	\$-1,000	\$12,000	\$28,000	\$34,000
Part 2. Net Present Values (NPV) for Each Period					
PV index @11%	1.00	0.901	0.812	0.731	0.659
NB present value	\$-10,000	\$ 8,108	\$10,551	\$11,699	\$ 3,952
Cumulative NPV	\$-10,000	\$-1,892	\$ 8,659	\$20,358	\$24,311
Part 3. Return on Investment (ROI) Example					
PV index @101.37%	1.00	0.4966	0.2466	0.1225	0.0608
NB present value	\$-10,000	\$ 4,469	\$ 3,206	\$ 1,959	\$ 365
Cumulative NPV	\$-10,000	\$-5,531	\$-2,235	\$ -365	\$ 0

and benefits in additional rows. With payback period as a measure of merit, the first section of the table is all the team may produce.

With an NPV measure of merit, the team may wish to add Part 2 of Table 9.2. The first row shows the present value index for an assumed 11 percent interest rate (the organization wants to earn 11 percent in its investment in the project) using the formula discussed earlier: $1/(1 + i)^n$. The client needs to provide the cost of capital or interest rate for use in the calculations. The present value row is calculated by multiplying the net benefit shown in Part 1 by the PV index for 11 percent in Part 2. The sum of all the present values to date is the NPV up through the period. For an assumed life of four years, the NPV of this proposed system is \$24,311.

Given the uncertainty of the cost, benefit, and lifetime estimates, the team may wish to state the NPV as around \$20,000 or \$24,000. Using the number \$24,311 implies a level of precision that almost certainly is misleading. As noted, the NPV number takes into account the net benefit flows, the effect of interest and the life of the project. NPV may prove a better measure of merit than payback period for projects with longer life. However, the longer the life of the project the higher the risk that something unexpected may happen to invalidate the NPV calculations.

Part 3 of Table 9.2 shows that this project yields a high internal rate of return or return on investment. An interest rate of 101.37 percent results in a net present value of zero at the end of year four. In other words, the project earns an average compound return of 101.37 percent on the invested capital for each year of the four years. The team may want to refer to an ROI of around 100 percent or an ROI much larger than the criterion of 30 percent. If the system collapses or is no longer needed after a couple of years or the net benefit estimates are too high, the ROI and the NPV will be less.

Features Analysis

When the total cost of a project is relatively small, usually within the signature authority of the client, many managers believe that detailed economic justification is excessive and unnecessary. Features analysis focuses on the features of a product rather than the economic benefit of the solution. The manager makes a list of **mandatory features** and a list of **desirable or optional features**. The team rates each alternative on how well it supports these features. Any alternative that meets the mandatory requirements and satisfies enough of the desirable ones is acceptable. This approach seems reasonable when the cost and risk are relatively small and the benefits intuitively obvious. Features analysis is discussed in more detail and is illustrated in a GB Video example in Chapter 10.

AN EXAMPLE OF ALTERNATIVES

Figure 9.2 shows an example of alternatives for the GB Video project. Note that the description of each alternative contains a section on evaluation. The evaluation measures for each alternative form the basis for the evaluation comparison discussed in the next section.

FIGURE 9.2 Alternatives for GB Video**Alternatives**

Based on discussions with Mr. Cosier, the team identified the set of alternatives shown below. The team has defined and evaluated each of the alternatives. Before performing in-depth analysis and evaluation of the four alternatives, the team conducted a feasibility study to determine if one or more of the alternatives fail to meet an important feasibility constraint. This analysis indicated that Alternative 1: Improve the Current Manual System and Alternative 3: Contract for Service do not meet constraints and should be dropped from further analysis for reasons discussed below.

The team conducted an in-depth evaluation of Alternative 2: Procure a Package System from a Vendor and Alternative 4: Contract for a Custom Package. Because Mr. Cosier provided estimates of costs and benefits, the team applied cost/benefit analysis, identified advantages and disadvantages, and examined the level of risk. In accord with Mr. Cosier's preferences, the team calculated the payback period for these two alternatives.

Alternative 1. Improve the Current Manual System. Based on a preliminary review, the team believes that several low-cost changes (probably less than \$5,000) to the current manual system could bring immediate improvement at GB Video. The team estimates that these changes would achieve about 40% of the benefits mentioned by Mr. Cosier or a cost reduction of about \$20,000 a year giving a payback period of 3 months. The risk associated with this option is low. The primary risk is that the changes may not result in the expected cost savings, but even if the savings do not happen, the initial cost is less than \$5,000.

When this option was discussed with Mr. Cosier, he repeated his statement that he does not wish to continue with the current system unless all computer-based alternatives are outside his payback criteria and his \$200,000 up-front cost limit. Since the team found other alternatives that will provide the desired functionality within the constraints, the team did not give further consideration to improving the current manual system.

Alternative 2. Procure a Package System from a Vendor. Package systems are software or computer program packages provided by third-party vendors. The team identified four vendors that sell or lease packages for video rental and return activities. After preliminary examination of specifications and demonstrations provided on the vendor Web sites, all four packages appear to meet the features requested by the client and the specifications derived by the team. The packages run under the NT operating system on any Intel chip server. The packages will run with MS Access, MS SQL server, or Oracle Server databases. The vendors claim that the packages can be installed and in operation in three months.

Advantages of package systems for GB Video include

- Meet the proposed system conceptual specifications.
- Available for implementation in less than a month.
- Relatively inexpensive and within the client's total investment constraint of \$200,000.

- Tested, proven functionality and performance.
- Relatively low risk.

Disadvantages include

- May require some organizational changes.
- Dependent on the vendor for maintenance and upgrades.
- Fixed yearly cost for upgrades independent of whether or not GB wants or needs the upgrade.

The team analyzed the costs associated with this alternative. Package costs appear to run around \$15,000 for initial purchase with use at the three GB Video existing stores. The vendors provide yearly maintenance and upgrades at around \$2,000 a year. The team estimates that installation including data conversion and training will cost \$10,000 and that hardware purchase costs for the three stores will total \$40,000. Hardware maintenance is estimated at \$1,000 a year.

The team examined the possible cost reductions from an automated system and concluded that Mr. Cosier's estimate of \$50,000 a year appears reasonable. The possible benefit of a 5% increase in sales is more difficult to analyze. Sales should increase by at least 5% for the existing stores and should lead to a more than 5% increase in profits. If Mr. Cosier opens additional stores as planned, the profits may increase by a significantly larger amount. To be conservative, the team used a 5% increase in profits for the existing stores or \$6,650 ($\$133,000 \times .05$).

The table below shows a summary of costs and benefits for purchase of a package.

Cost/Benefit Summary for Purchase of a Package

	Initial	Year 1	Year 2	Year 3
Costs				
Computer hardware	\$40,000	0	0	0
Package cost	\$15,000	0	0	0
Data and training	\$10,000	0	0	0
Hardware maintenance	0	\$1,000	\$1,000	\$1,000
Package support	0	\$2,000	\$2,000	\$2,000
Benefits				
Cost reduction	0	\$50,000	\$50,000	\$50,000
Profit increase	0	\$6,650	\$6,650	\$6,650
Net benefit	(\$65,000)	\$53,650	\$53,650	\$53,650
Cumulative net benefit	(\$65,000)	(\$11,350)	\$42,300	\$95,950

If everything goes as planned, GB will recover its investment early in year 2 for a payback period of about 15 months ($12 + [11350/53650 * 12] = 14.5$ months). Even if profits do not increase at all, the payback period of 17 months still meets the 2-year payback constraint set forth by Mr. Cosier.

Alternative 3. Contract for Service. The team searched for possible ASP vendors who might submit a bid to perform the functions identified in the conceptual specifications for the GB Video system. The team contacted a selection of current ASP vendors and also talked with several large video rental companies to find possible bidders. The video companies declined to bid. One said that providing such a service to help a potential competitor was not in their best interests.

None of the ASP vendors currently provide a video rental system and none expressed any interest in buying one of the available video packages and gearing up to offer such a service. Several said that they did not think a viable market existed. The team did find one ASP, Integrated Computer Services, that would develop a video rental package and supply the service to GB Video via an Internet interface. ICS estimates that developing, testing, and installing the program at GB will take 18 months. ICS proposes that GB Video pay for the initial development, setup and marketing of the service, estimated at \$500,000 and then share in the profits of subsequent sales of service to other video rental companies. ICS estimated that GB Video would recover its investment in 4 years and could make significant profits after that time. Mr. Cosier rejected this option as outside of the business plan for GB Video, high risk, and an unacceptable payback period.

Alternative 4. Contract for a Custom Package. The team identified a local software company, OkieComp, which offered to submit a bid to write a software package to detail design specifications provided by GB Video. OkieComp currently sells a similar well-regarded package developed for equipment rental stores. OkieComp stressed the company provides programming services, not system design services. GB must give OkieComp that detail design specifications for the program and specify the infrastructure in which the programs will operate. OkieComp will write the software and deliver an extensively tested package to GB Video. OkieComp initially estimated a total initial cost of \$99,000 to develop, test, and install the system including hardware. Based on this estimate, Mr. Cosier asked the team to proceed with further exploration of this alternative.

The team developed detail design specifications and incorporated them in an RFP and Mr. Cosier sent the RFP to OkieComp. OkieComp's bid contained the following conditions.

1. The initial cost for developing, testing, and installing the software and hardware that fully meets the specifications provided by GB Video in the RFP is \$94,500. The hardware list appears in Appendix A of the bid. The amount of \$94,500 will be paid at the time that GB accepts the software.
2. OkieComp will deliver the software 6 months after receiving a firm contract. OkieComp will work with GB Video to help GB Video demonstrate that the software works according to specs. OkieComp will correct any noncompliance issues identified by GB at no cost for 5 years.

3. If GB Video wishes to make any changes to the software or infrastructure, OkieComp will charge GB the actual labor cost to OkieComp plus 70 percent to make, test, and install changes to the software.
4. OkieComp will load the GB data into the new system and train the people at GB for a cost of \$5,000.

Advantages of OkieComp include:

- Product will perform exactly as specified by the team and client.
- No organizational changes required.
- No payment due until the product meets acceptance tests—reduces the risk of damage to GB if OkieComp is unable to deliver.
- GB controls upgrades.
- Five-year guarantee.

Disadvantages of OkieComp include

- Possible risk that OkieComp cannot deliver estimated as small, and GB has some protection.
- Possible risk that OkieComp will go out of business estimated as medium.
- Upgrade costs not fixed.
- Three months or more required to obtain the software.

The costs for this alternative appear in the OkieComp proposal. As in the package system alternative, GB can obtain hardware maintenance from a third party for \$1,000 a year and the benefits remain a cost reduction of \$50,000 and additional profits of \$6,650 per year.

The table below shows a summary for the OkieComp alternative.

OkieComp Cost/Benefit Summary

	Initial	Year 1	Year 2	Year 3
Costs				
Package and hardware	\$94,500	0	0	0
Data and training	\$5,000	0	0	0
Hardware maintenance	0	\$1,000	\$1,000	\$1,000
Package support	0	0	0	0
Benefits				
Cost reduction	0	\$50,000	\$50,000	\$50,000
Profit increase	0	\$6,650	\$6,650	\$6,650
Net benefit	(\$99,500)	\$55,650	\$55,650	\$55,650
Cumulative net benefit	(\$99,500)	(\$43,850)	\$12,200	\$67,850

If everything goes as planned, GB will recover its investment late in year 2 for a payback period of about 22 months ($12 + [43850/55650 * 12] = 21.5$ months). This alternative appears to meet all the 2-year payback constraints set forth by Mr. Cosier but it probably involves more risk than the package alternative. If profits do not increase, the payback period is slightly more than 2 years. If OkieComp is unable to deliver on time or at all, the risk to GB is small. GB can continue to use the existing manual system and can purchase a package system if needed.

Page 5

THE EVALUATION COMPARISON AND THE RECOMMENDATION

At the end of the alternative evaluation process, the team has derived costs, benefits, risk, and other measures, including perhaps, a list of advantages and disadvantages, for each alternative. A typical client may find it tedious and confusing to look back and forth through the alternatives to compare them. The team can simplify and improve the comparison process by putting the evaluation data in a form that facilitates comparison. A good format is an **evaluation summary table** with a row for each alternative as shown below in Table 9.3.

The Table 9.3 format can summarize the evaluation information. The key idea is to present the essential data in a form that is easy to see and compare. The specific features included in the table will vary with each project. The team includes the features that are important to the client and the organization. The table should reflect the performance measures identified earlier for the organization. An actual table may have more feature rows and might have a paragraph describing risk in place of one word. An illustrative evaluation summary table for GB Video, for example, appears in Figure 9.3. The summary table follows the rules for all summaries: It presents only information that is discussed in the evaluation of each alternative. When new, previously undiscussed information suddenly appears in a summary table, the new material may both confuse and annoy the client.

The Recommendation

Normally, the recommendation, if present, follows the evaluation summary. Some clients do not want the team to make a recommendation. In this event, the

TABLE 9.3
An Evaluation
Summary

Feature	Alternative		
	1	2	3
Description	Current system	Contractor	Build in-house
Development cost	\$0	\$50,000	\$40,000
Payback period	———	14 months	16 months
Risk	Medium	Low	High
Meets constraints	Yes	No	Yes

FIGURE 9.3 Evaluation Comparison and Recommendation

The team defined and evaluated four alternatives for the GB Video Rental System:

1. Improve the current manual system.
2. Procure a package system from a vendor.
3. Contract for service.
4. Contract for a custom package.

The table below summarizes the results of the evaluations of the alternatives. The features listed in the evaluation table are ones that the client mentioned or stressed.

Evaluation Summary Table for GB Video

Features	Alternative			
	1	2	3	4
Description	Current	Package	ASP	Custom
Client preference	Low	High	Low	Possible
Improves performance	Some	Yes	Yes	Yes
Meets client constraints	No	Yes	No	Yes
Initial cost	\$5,000	\$65,000	\$500,000	\$99,500
Estimate payback	3 months	15 months	48 months	22 months
Meets specs	70%	95%	100%	100%
Client controls updates	Yes	No	Unclear	Yes
Risk	Low	Low	High	Medium
Time until operational	0	3 months	18 months	6 months
Custom modifications	Yes	No	No	Yes

As noted earlier, the client and team agreed not to explore Alternatives 1 and 3 in depth. The data on them is presented here to provide prospective. All of the alternatives except the current system contribute to GB's performance objectives, i.e., reduce cost by \$50,000, provide support for additional stores and increase profits. Alternatives 2 and 4 meet the constraints set forth by the client. While several packages meet the mandatory specifications, none of the packages contain all of the features desired by the client.

Recommendation

The team recommends Alternative 4: Contract for a Custom Package. After carefully reviewing the summary evaluation table, the team concludes that the custom package alternative does the best job of meeting the client's needs. The solution meets the constraints of a 2-year or less payback and an initial investment of \$200,000 or less.

While the custom package has a longer payback period and higher initial cost than the off-the-shelf packages, it offers a number of advantages. It contains 100% of the features desired by the client and thus does not require the client to change practices and procedures to fit the package. The vendor also agrees to modify the package when and as requested by the client.

The risk of this alternative is limited by the provision that GB does not pay for the package until it is tested and accepted as operational. If for some reason, OkieComp fails to deliver, GB can purchase an off-the-shelf package and have it operational in 3 months. The major risk to GB is that OkieComp may go out of business and not correct problems or make modifications if desired. During an informal review, both the team and Mr. Cosier concluded that the risk is acceptable and that the advantages of the custom package outweigh the disadvantages.

Page 2

team provides the evaluation of alternatives and the client selects the solution. In most cases, the team, often after coordination with the client, does make a recommendation. A good recommendation is clear, specific, and *consistent with the evaluation summary*. The recommendation focuses on facts and avoids unsubstantiated rhetoric. Telling the client that the recommended solution is “optimal” probably misstates the facts. A better statement is simply, “The team recommends solution X because—(discussion of the evaluation table results).” Guarantees of results and trouble-free implementation also probably represent misstatements. A better alternative is a clear explanation of possible risks or problems as identified in the summary table and the contingency plan to deal with them

On the other hand, teams should “sell” their recommendations with a clear and effective statement of the merits in the context of the client. The team can emphasize what the client thinks is important and explain both why the selected alternative is recommended and why the others are rejected. The team should try to determine the client’s solution preferences before the final presentation. If the client rejects or redesigns the recommended solution during the formal presentation, the team will look unprofessional and poorly prepared. If the team recommends a solution other than the lowest cost or highest net benefit solution, the team should take special care to justify the action. For example, in their recommendation, team members might state, “The project team recommends the use of a contractor to fix the problems in the current system as outlined in alternative 2. While the contractor solution costs 25 percent more than an internally developed system, the contractor solution is forecast to provide larger benefits and thus has a slightly shorter forecast payback period. In addition, the contractor has extensive experience in the area and a good history of performance on similar projects for other companies. As a result, we believe the risks of cost overruns or delays in completion are lower with the contractor alternative.”

An illustrative example of an evaluation summary table and recommendation appears in Figure 9.3. Note that the table follows the general rule for a summary, that is, it uses the information from the alternatives discussion in Figure 9.2. The recommendation, in turn, builds on the data in the summary table.

Client Approval to Proceed

The decision to pursue in more depth one of several alternatives represents a critical managerial decision point. While the team holds the primary responsibility for developing and evaluating alternatives, the client holds the primary responsibility for approving the selection of the specific alternative solution for system delivery. Before proceeding, the team should present the evaluation and recommendation to the client and obtain **client approval**. The recommendation asks the client for approval to continue the project, not necessarily to complete it. The client may wish to include additional decision points later in the project. To make an intelligent decision the client needs to know:

- The evaluation and recommendation information.
- The cost and time for completing the next step (system delivery or a first part thereof).
- A revised estimate of (1) the cost and (2) the time schedule for the project.

While informal discussion may suffice, the team may wish to prepare a memorandum for the client's signature that puts in writing the above information.

Summary

Evaluation and alternatives remain central themes throughout the life cycle of a system. The team begins the evaluation process with the strategic analysis of the organization during project definition. Once the conceptual specifications for the proposed system are known, the team may identify or construct several alternatives, examine feasibility, evaluate in-depth the feasible ones and select one, or perhaps several, alternatives that appear to provide satisfactory solutions as the recommended solution.

This team uses the strategic framework to develop and evaluate meaningful alternatives for a proposed system. Today, many successful IS groups focus their effort on locating and exploiting applications that directly impact the performance of the company. The effectiveness of an IS application may be measured by how it works to improve the overall performance of an organization. Finding the recommended solution involves four key areas: the mission, the goals and objectives, the indicators of performance, and the impact on performance of the solution.

Alternatives include both feature and sourcing options. Features consist of functionality, performance, and infrastructure. The team explores different levels of functionality and infrastructure with the resulting levels of performance. Sourcing options include in-house development and such outsourcing options as packages, contract development, and application service providers (ASPs). The team provides a full description of each of the alternatives including an evaluation of the alternative.

Evaluation of alternatives is complex. The goals of evaluation are (1) to avoid disasters; (2) to focus on promising courses of action; and (3) to find a satisfactory or good solution. Feasibility analysis can help the team to avoid disasters and eliminate unpromising alternatives from further consideration. A key consideration in evaluating alternatives is risk. Risk may arise from many areas including staff, client, vendor/architecture, and competition. Risk can lead to disasters. Risk-adverse clients want the team to consider the lower risk alternatives.

The more promising alternatives are candidates for in-depth cost/benefit analysis and/or features analysis. Cost/benefit analysis consists of three steps: identify costs and benefits for each alternative, select an evaluation metric, and rank alternatives. Applying such evaluation metrics as net present value (NPV), return on investment (ROI), and payback period may identify one, several, or no satisfactory solutions. Cost/benefit tables can help teams organize, examine, and display the data for each alternative.

An evaluation summary compiles all evaluation data in a form (often a table) that allows for a quick and thorough comparison of data forming the basis for the recommendation. The evaluation summary follows the rule for all summaries: all of the data in the summary must appear in the discussion of the alternatives. A good recommendation is clear and specific and made in conjunction with the client. The data in the evaluation summary must support or be consistent with the recommendation.

The recommendation represents a key decision point. When the client accepts the recommendation, the team enters into the system delivery phase. The team begins to procure a system from a vendor, to build a system, or some combination. In a few cases, the client may wish for the team to explore both build and buy options in depth. The team may build a prototype that contains the exact functions the client wants and also find packages that match as closely as possible with the client's requirements. The team can demonstrate both the prototype and the packages and then let the client make a final choice. However, in many cases the cost of detail design for multiple alternatives is too high. Evaluation, along with most things in systems work, involves compromise.

Key Terms

alternative, 300	desirable or optional features, 328	interoperability, 307
application service provider (ASP), 306	evaluation, 309	legal feasibility, 312
availability, 308	evaluation metrics, 319	mandatory features, 328
break even analysis, 319	evaluation summary table, 329	net present value (NPV), 320
contract	feasibility analysis, 311	one-time costs, 318
development, 306	features analysis, 311	ongoing costs, 318
cost avoidance, 317	implied benefits method, 322	operational feasibility, 312
cost/benefit analysis, 311	intangible benefits, 317	outcome feasibility, 313
cost displacement, 317	internal rate of return (IRR), 322	package system, 305
cost reduction, 316		payback period, 319
design option, 300		performance, 308
		recommendation, 299

response time, 308	risk analysis, 313	sourcing option, 302
return on investment (ROI), 322	risk reduction, 317	technical feasibility, 312
revenue and performance enhancement, 317	schedule and cost feasibility, 312	technology and vendor risk, 313
	security feasibility, 313	total cost of ownership, 318

Review Questions

Answer the following questions regarding these topics.

1. Alternatives.
 - a. Why should a team not explore all possible alternatives in depth?
 - b. How does the sourcing approach affect possible systems features?
2. Design options.
 - a. What are the design options that a team should consider?
 - b. Why should a team always consider the current system as an alternative?
 - c. When is zero-based design appropriate?
3. Functionality. Why might a legitimate alternative not have all of the features defined in the proposed system analysis?
4. Sourcing.
 - a. What are key indicators that a solution should be built in-house?
 - b. What are key indicators that a solution should be outsourced?
5. Outsourced solutions.
 - a. What is the difference between hiring contractors to help build a system and contracting for a solution?
 - b. Give an example of when each of the following sourcing options would be desired: in-house development, contract development, package systems, or application service providers.
6. Performance.
 - a. What are some standard measures of performance of a system?
 - b. What are the basic questions that a team should ask to address strategic or tactical requirements for the system?
7. Feasibility analysis.
 - a. What is a feasibility analysis?
 - b. What are each of the following types of feasibility: schedule, cost, technical, legal, operational, outcome, and security.
 - c. What are overriding considerations that would make a new system essential in spite of feasibility difficulties?
8. Recommended alternative. What are some factors, in addition to cost, that the team should consider when choosing a recommended alternative?
9. Risk evaluation.
 - a. How do you handle risk in evaluating alternatives?
 - b. What are some of the factors that make a project risky?
 - c. How can you reduce risk in a project?
10. Cost.
 - a. Distinguish one-time costs, ongoing costs, and operational costs.
 - b. What are the primary sources of cost in a system?

11. Benefits
 - a. What are some of the key difficulties in evaluating benefits.
 - b. List the primary sources of benefit for a potential system. Give an example of each one.
12. Evaluation.
 - a. What are the basic cost/benefit evaluation metrics that are normally used?
 - b. What are the strengths of each one?

Critical Thinking Exercises

Individual Exercises

1. Your client wants a new system to handle payroll in the company. Which sourcing option would you choose? Explain your answer.
2. Give advantages and disadvantages associated with the different sourcing options.
3. For the data in the table below, solve the following:
 - a. Find the net benefit, cumulative net benefits, and payback period.
 - b. Calculate the NPV at the end of year 4.
 - c. Calculate ROI at the end of year 4.

Year	0	1	2	3	4
Costs	\$20,000	\$10,000	\$ 8,000	\$ 9,000	\$ 9,000
Benefits	\$ 0	\$12,000	\$15,000	\$18,000	\$18,000

Group Exercises

1. Your client needs a system to keep track of inventory. Inventory is currently tracked by manually counting and entering the amounts at the end of each day.
 - a. What are the alternative solutions?
 - b. Write a description for each alternative solution.
2. What is your recommended solution for the inventory system in Group Exercise 1? Justify your answer.
3. **City Alarm billing system.** You have been asked to recommend a billing system for City Alarm company. The company monitors burglar alarms for businesses and residences in the city and bills clients on a monthly, quarterly, or annual basis, depending on the customer. Your client, Bill Smith, identified two essential requirements:
 1. The system must be able to print statements by month, quarter, or year.
 2. The system must be able to determine the age of accounts so that bills more than 30 days late can be charged interest.

In addition it would be desirable for the system to be easy to use and to print envelopes in zip code order.

Currently, City Alarm prepares bills by hand. Bill Smith estimates that it takes his bookkeeper, who earns \$20 per hour, about 75 hours per year to prepare bills. Of that time, 50 hours is spent in organizing accounts and preparing labels. With the current system, City Alarm mails first class. Sorting mail in zip code order would save 10 cents per envelope for each of the 1,000 envelopes City Alarm mails each month.

Billing-dot-com. Billing-dot-com is a company that has a billing system for alarm companies that sells for \$1,000. The software will also require a new computer system

at a cost of \$2,000. The product has an annual maintenance fee of \$500 per year. City Alarm estimates that it will take about 20 hours to train the bookkeeper.

- a. Calculate the payback period for Billing-dot-com's product. Show your work in a table. List sources of cost and sources of benefit separated by startup and ongoing values by year.
- b. Calculate the net present value for this system assuming a life of eight years.
- c. Calculate the return on investment assuming a life of eight years.

References

- Atkinson, Anthony, et al. *Management Accounting*. Upper Saddle River, NJ: Prentice Hall, 1995.
- Brigham, Eugene; and Louis Gapenski. *Intermediate Financial Management*, 5th ed. New York: Dryden Press, 1996.
- Emery, Douglas; John D. Finnerty; and John D. Stowe. *Principles of Financial Management*. Upper Saddle River, NJ: Prentice Hall, 1998.
- March, James G.; and Herbert A. Simon. *Organizations*. New York: Wiley, 1958.

System Delivery

System delivery brings major challenges for both the clients and the team. The team schedules and carries out such activities as:

- Preparing the final logical and physical design specifications for the proposed system.
- Purchasing or building the production system and/or a proof of concept model.
- Procuring any required additions or modifications to the physical infrastructure.
- Carefully and completely testing the system.
- Implementing the system or preparing an implementation plan.
- Preparing plans for maintenance and a post-implementation audit.

Many of the preceding activities can involve major expenditures of money. Often the bulk of the total project costs occur during system delivery to pay for people, software, hardware, testing, data conversion, and training. Implementation expands the group of people involved with designing the system to a possibly much larger group of people who will use or be impacted by the system. When the new system starts to operate, the system also will have an impact on the organization ranging from highly beneficial to disastrous. For these reasons, the client often closely follows the system delivery process and may wish to have input on a number of the decisions.

At the end of the proposed systems phase, the team made a major decision: build or buy the new system. With a buy decision, the team can prepare the necessary analysis and documentation to outsource following the guidance in Chapter 10. With a build decision, the team can follow the guidance in Chapter 11 to convert and expand the conceptual specifications into detailed logical and physical design specifications. All teams may perform some or all of the activities in Chapters 12 and 13: obtaining and using a proof of concept model, testing the system, and preparing implementation plans.

Chapter 10 explores the issues associated with outsourcing. While outsourcing often gives the client a beneficial option, outsourcing can bring far less desirable results than expected unless managed carefully. Before entering into an outsourcing contract, the team needs to understand a broad set of issues

Chapter Ten

Outsourcing

Chapter outline

Introduction

The Outsourcing Process

- Models of Outsourcing*
- Determining Requirements*

Product Features

- Functional Features*
- Process Features*
- Data Features*
- Learning from Products*

Operational Features

- Performance*
- Usability*
- Interoperability*
- Security*
- Maintainability*
- Software Licenses*
- Costs and Prices*
- Laws and Regulations*
- Product Flexibility*
- Contract Compliance*
- Organizational Fit*

Vendor Roles

- Vendor Features*
- Vendor Stability*

- Vendor Support*

- Delivery Record*

- Product/Vendor Selection Issues*

Request for Proposal

- RFP Content*

- GB Video RFP Example*

Features Evaluation

- Ranking Methods*

- Identifying Candidate Solutions*

- Assigning Ratings*

- Outcomes*

- GB Video Example of a Weighted Features*

- Analysis*

- Contracts*

Summary

Key Terms

Review Questions

Critical Thinking Exercises

- Individual Exercises*

- Group Exercises*

INTRODUCTION

The paths of information system solutions resemble a tree. During project definition, most teams follow a similar path, the trunk of the tree. As work proceeds, the nature and requirements of different projects take teams along diverging branches. During the evaluation process discussed in Chapter 9, the team selects a sourcing option—either to build or buy. As a result, in the system delivery stage, teams may follow quite different paths. For example, some teams will build or develop detailed specifications to build a system while others will explore an **outsourcing** option. Similar differences arise from the nature of the solution. A Web design project may require a different set of activities than a data warehouse project. The chapter covers the activities for an outsourcing solution—purchasing or leasing an application or service from a vendor. The activities for building a new application system solution appear in Chapter 11.

The process for outsourcing begins at the same place as building a system: the conceptual requirements. As discussed in Chapter 8, the team develops a set of conceptual requirements or specifications for a new application. While building a system focuses on identifying and coding detail logic to meet the requirements, outsourcing focuses on identifying vendors and/or products, and evaluating features of products against requirements. If the clients insist that a solution must track their desired requirements exactly, then outsourcing may offer a poor alternative. Outsourcing may require some level of compromise in adjusting the existing organization and procedures to the functionality of the purchased product.

The inputs to the outsourcing task include:

- The project definition materials.
- The statement of work (SOW).
- The conceptual specifications for the proposed solution.
- The evaluation that led to selecting the outsourcing option.

Possible outputs from the outsourcing function include:

- An outsourcing relationship model, such as commodity or association.
- A features comparison matrix for vendors/product combinations.
- An evaluation metric.
- A request for proposal (RFP) document.
- An identification of possible vendors and products.
- An evaluation of vendors and products.
- A recommended solution.
- A demonstration of the recommended solution.

Outsourcing comes in a variety of forms. An outsourcing product may consist of the purchase or lease of a package application and/or a contract for developing an application or providing a service. A common approach to evaluating desired features of products and vendors involves the use of a features matrix as described in the Features Evaluation section of this chapter.

THE OUTSOURCING PROCESS

As noted earlier, the field project activities for purchasing a solution differ significantly from the activities for building one. In place of creating design documentation and building a prototype, the team that follows an outsourcing option may execute some or all of the following sequence of steps:

1. *Select an outsourcing model.* The team selects an outsourcing model that determines the relative importance of product specifications and the relationship with the vendor.
2. *Refine the requirements.* The team examines, revises, details, and expands the conceptual specifications developed in Chapter 8 for the proposed system to specify the logical and physical specifications for the new system. Normally the team will prepare narrative and graphical models as appropriate that describe the desired features and may note why the client wants or can benefit from the feature.
3. *Build a **comparison matrix**.* The team determines the mandatory and desirable functions and features for the proposed product and vendor and decides upon an **evaluation metric**: economic value, rating points, or a features list. The team presents these results in a **features matrix**—a matrix containing a quantitative or qualitative value for each feature of each product by vendor.
4. *Specify a rating process.* The team specifies the process to assign the value or rating for each entry in the features matrix. Possible approaches to obtain ratings include reading published reviews, surveying users, hiring an expert or panel of experts, or using a combination.
5. *Write an RFP.* Many organizations solicit bids for a solution through a formal **request for proposal (RFP)** or a request for quote (RFQ) process. The team may prepare a request document that includes contract terms, solution requirements, bid content, and evaluation methods.
6. *Collect potential solutions.* The team may search for and identify vendors, send promising vendors the RFP, and receive bids from interested vendors.
7. *Evaluate potential solutions.* The team may conduct an evaluation of the bids with the methodology specified in the RFP.
8. *Select, justify, and demonstrate a recommended solution.* When appropriate, the team selects and demonstrates the recommended solution to the client.

Models of Outsourcing

With all outsourcing options, the team must decide on an appropriate framework or model for the outsource action: a **commodity model** or an **association model**. Table 10.1 shows the key features for each model of outsourcing.

A commodity product remains essentially the same, independent of the vendor who provides it. Internet access and personal computers in the last few years have moved close to becoming commodity products. While differences exist among the products supplied by different vendors, the primary outsourcing decision focuses on the price for the desired functionality rather than on a long-term

TABLE 10.1
Models of
Outsourcing

Outsourcing Model	Product	Management Focus
Commodity	Specified features as delivered	Price and performance levels for the product
Association	Specified features as delivered plus future support including features, not yet known, in upgrades	Price and performance levels for the product plus the characteristics of and the relationship with the vendor

relationship with the vendor. With commodity outsourcing, the team focuses on defining the desired features and developing methods to measure them.

With commodity agreements, the purchaser tries to negotiate the best available price for the desired feature set and verifies that the vendor provides the agreed-upon performance. The ongoing process for a commodity purchase requires identifying and tracking the critical features of the product to assure that the features in the vendor's specification work as described and/or that the vendor meets the contracted service levels. Commodity contracts for services require careful development of service level agreements that specify such things as availability, response, capacity, and quality.

Association outsourcing assumes that the vendor and the customer will enter into an ongoing, mutually beneficial relationship. A successful association consists both of establishing clear expectations and working out problems in a mutually satisfactory way. Price remains important, but the initial purchase or contract price may represent only a small fraction of the total life cycle cost for the application. Serious problems can arise when the vendor and customer hold different expectations or when one party tries to benefit at the expense of the other. For purchases in the association category, the team wants to examine such issues as vendor stability, vendor delivery performance, and ongoing vendor support including upgrade plans. Association contracts benefit from careful attention to dispute resolution procedures and personal relationships.

For example, a multimillion dollar acquisition of an enterprise resource planning (ERP) system focuses heavily on the features of a long-term relationship with the vendor. The purchaser wants to build a relationship with an ERP vendor who can supply a useful product over a number of years. The purchaser wants to deal only with ERP vendors who the purchaser forecasts can and will maintain, support, and upgrade their ERP product on an ongoing basis. The expected total costs and benefits of ownership probably will reach more than 20 times the ERP package purchase price and may vary from vendor to vendor by much larger amounts than the difference in package prices. In these cases, the nature of the ongoing relationship generally holds more importance than the package price. As a result, the team may spend as much or more time and effort analyzing the vendor's characteristics as the vendor's product.

Many and probably most products, for example, a typical small package system, fall in between the commodity and association classes. With a small package procurement, the team may spend much of its time evaluating the features of the product, but the relationship with and features of the vendor also are important.

Determining Requirements

Once the team and the client come to an understanding on the appropriate outsourcing model, the team refines the specifications in sufficient detail to evaluate the suitability of various products and vendors. Clearly, the more expensive the contract, the more detail is warranted. The team determines the set of mandatory and desirable features for the product/vendor combinations. A purchased product is the initial thing that the vendor provides to the client; a package program or a service and product features describe the package or service. Vendor features describe such vendor characteristics as stability and delivery record. The output of this features **requirements** work normally appears in a narrative that includes one or more paragraphs for each feature. For a complex, expensive product these discussions may take tens or even hundreds of pages. For a typical field project with products of modest complexity, a discussion that covers a couple of pages probably will suffice.

As noted in Chapter 9, features fall into several groups depending on how important or essential the features are to the problem the client wishes to solve. **Mandatory features** are ones that the client believes must exist in a solution. A mandatory feature either is or is not present in a solution. If 80 percent of a mandatory feature is included, the feature is not present. One rule to identify a mandatory feature says, *If the product/vendor solution does not contain the mandatory features, the client may decide to reject the solution from further consideration.* When all available and/or affordable solutions lack one or more mandatory features, clients sometimes revise their mandatory requirements.

Desirable features are ones that the client wants, but the client may decide to accept a solution without them. Desirable features can be present in part in a solution. A solution might have 80 percent of the functionality that the client considers to be a desirable feature. When a solution lacks some or all of a desirable feature, the organization can work around the missing feature in a reasonable fashion. A client may accept a solution that lacks part or all of one or more desirable features if the solution brings enough value with the features that are included.

The client should rank the relative desirability of each desired feature. The team can ask the client to generate a dollar value for each desired feature, but often the client will find it difficult to estimate dollar values. Other simpler alternatives include asking the client to rank order the desirable features or to assign a measure of desirability—for example, high, medium, or low or a number measure on a scale of 1 to 10. These rankings allow the team and client to focus attention on the most valuable of the desirable features rather than on a long list.

“Nice to have” features are desirable ones that seem interesting and possibly useful but do not relate directly to the client requirements. A number of the features included in a product by the vendor may fall in the nice to have category. Sometimes the client may suggest some nice to have features, but more often the client lists them as desirable features with a low ranking. In most cases, the team omits nice to have features from the features matrix and simply lists them as other included features for the evaluation.

PRODUCT FEATURES

Product features include functionality and operability. **Functional features** describe what the product must do, the functionality of the product. **Operational features** describe how the product works. Operating features can include options, maintainability, architecture, and other properties that affect how the product will operate.

Functional Features

Determining functional features or functions for a product starts with a review of the conceptual requirements that the team prepared for the proposed solution (see Chapter 8). The team may review the proposed system narrative and graphical data and process models to identify these functional features.

Process Features

Process Features often appear as part of a hierarchy illustrated graphically by data flow diagrams or function hierarchy diagrams. The boxes on the first explosion DFD for the proposed system probably represent high-level processes that the client will recognize and understand. These high-level processes may contain a number of subprocesses as shown in the DFD explosions.

Data Features

The conceptual entity relationship diagram (ERD) specifies the required **data features**. The team generally can take the graphical data and process models and the metadata for the proposed system and incorporate this information directly into a requirements narrative for the purchased product or service.

Learning from Products

The team also should determine the functionality contained in a representative sample of available products and review the available functionality with the client. The client and the team may wish to revise the conceptual requirements after seeing the functions that are available in products. During the discussion with the client, the team can begin the process of classifying functions as mandatory, desirable, and nice to have. If desired, the team, at the same time, can begin assigning relative importance weights for each function. The clients decide the weights, but the team needs to set up a procedure to gather the information.

Operational Features

In addition to functionality, clients want to evaluate such operational features of products as performance, usability, maintainability, laws and regulations, interoperability, security, contract compliance, and cost. The team probably identified some of these features during the construction and evaluation of alternatives (see Chapter 9). At this point, the team further analyzes operational features to determine which ones to include in the narrative and the features matrix. The following material describes a number of areas the team may wish to include. For a specific product, the team may include all or only some of the

areas as relevant. In common with functions, the team will develop and apply a process for ranking the desirability of each operating feature.

Performance

Performance features measure how well or fast the product should run or operate and/or how large a database and transaction volume the product will handle. Typical performance measures include response times for interactive modules, retrieval times for data, processing times for transactions, and error or crash rates. Some packages that perform very well in environments with small files or limited transaction volumes may crash or run very slowly as file sizes and/or transaction volumes increase. For example, a Microsoft Access database may work well for a departmental system but perform inadequately for a larger unit. If the application will support multiple users, the team should determine how it handles multiple access and how many users it can support. Other relevant questions might include, Is it Web-enabled and is it scalable? Performance measures may depend on the total environment, including the package; the operating system, database engine, and other software; the network; the servers and other hardware; file and message sizes, and the usage pattern.

Usability

Usability or ease of use frequently appears among features desired by the client. Users may avoid, degrade, or circumvent a system that they consider difficult to use. Ease of use represents an ill-defined concept since “easy” depends on the person. Ease of use breaks into two parts: ease of learning for inexperienced users and efficiency for experienced users. For example, menu-driven systems are easy for beginners to use, but slow and annoying for expert users. Keyboard shortcuts are fast for expert users but difficult for beginners. A user interface that matches well with the tasks the users must perform can facilitate learning and use. Good design, good manuals, good help functions, and good training programs can contribute to ease of learning.

Efficiency for experienced users raises some complex issues. An efficient system allows the user to accomplish frequent or common tasks by selecting or activating one function; infrequently performed tasks may require using multiple functions. An efficient online interface for an experienced user allows the user to accomplish the needed tasks rapidly by such approaches as the following:

- Placing input entries in the same order as input naturally occurs in the task.
- Retrieving related blocks of data in place of making the user initiate multiple requests for data items, for example, retrieving all the customer data in place of separate retrievals for name, address, phone, and so on, or retrieving the quantities on hand for all the colors, not just for the color the customer requested.
- Reducing the number of keystrokes and mouse clicks required to accomplish the task.
- Reducing the need to switch between screens.

In some cases the team can find a review or find an organization that has used the product before. These sources can give helpful input on ease of use for the

product. If possible, the team also should obtain an actual or demonstration copy of the product and let selected users from the client's organization try it out.

Interoperability

Most systems in a company must work with other parts of the company operating environment. Since few systems really stand alone, the team investigates how well the different solutions will integrate with the existing environment—data, process, and infrastructure. **Interoperability**, the ability to work with other components and systems, plays a critical role in determining the costs and benefits of a new application. The team answers the following questions for each viable solution: With what operating system will it work? What database environment will support it? How will it communicate: web or network? How will it import and export data to other applications? What other technologies, for example, servers, terminals, or other, will it need? Will it work within the existing organization?

If the product requires components or systems not in place at the client's location, the team needs to access both the cost and feasibility of providing the needed environment. Many IT organizations develop shop standards that specify the allowed or supported environments. One shop may stress Microsoft operating systems; another may select UNIX. One group may understand and apply Visual Basic; another may use DreamWeaver. Products that cannot operate effectively within the IT standards pose procurement, training, skill, and support problems for the organization, and the client may deem them unacceptable. **Application service providers (ASPs)** also have technical environments that they support. While the ASP environment need not match the internal customer environment, the ease and effectiveness of interfaces between the customer and ASP represents a major operational feature.

Security

Security features prevent unauthorized access and actions. Unauthorized access or actions can result in theft of data or other resources, undesired or illegal modification of data, corruption of data, damage to the application code, or degradation of performance. Some access and action control may reside in the network; however, the team should determine what, if any, security functions should reside in the application or package. In the modern world of hackers, global communication, and potential terrorism, security features may deserve heavy weight in an evaluation. Protecting a system against internal and external threats is a full-time job for full-time professionals. Web systems, in particular, are especially vulnerable. Teams should consider recommending outsourcing for critical applications, particularly Web-enabled ones, for clients without a professional information system security staff.

Maintainability

Once an organization installs a product, someone—the client, the vendor, or a third party—must maintain the product. Most features matrices contain entries for one or more **maintainability** issues. Some products may offer easier or better maintenance arrangements than others. If the client decides to maintain the

product with internal staff, a first issue is access to the source code. In some cases, a vendor will not provide source code. Self-maintenance also requires that the organization either has people who can provide technical and user support for a product or can obtain such support from consultants or new hires.

A number of good products have pockets of usage in particular geographic areas or within specific business categories. Such products are much easier to maintain if there are a number of local professional experts available for permanent hire or as consultants. The team may wish to look at the local environment to determine the products that are in general use nearby. One client rejected a preferred accounting software package because no internal staff members or local consultants possessed the skills to maintain it. Many vendors offer maintenance themselves or through licensed or recommended third parties. When the client decides to use vendor or third-party maintenance, the issues become cost, availability, and quality as discussed under support.

Software Licenses

Package software typically sells or leases with some form of **software license** arrangement. The salesperson may quote the price for a license that provides far less service than the client requires. License terms deserve careful analysis and inclusion in the features analysis. While actual rules vary from license to license, typical number-of-copies provisions include:

- A single station license permits the installation of the software on one, and only one, machine at a time. Backup copies may or may not be permitted.
- A single user license permits the installation of the software on a single machine as well as on a home computer with the assumption that only one machine will be in use at a time.
- Network licenses permit the installation of the software on a network, usually with a monitor that limits the number of concurrent users. This license is the most common form for stand-alone network software.
- Site licenses permit the installation of up to a given number of instances of the software on any machine the organization owns at the discretion of the leasing company.

A second set of license issues defines the time duration the license. Typical provisions are as follows:

- Lifetime license. In a lifetime license, the purchaser receives a lifetime authorization to use the software. Free ongoing upgrades and service may be available for an initial period. Continuing support usually is on a fee for service basis or requires a separate maintenance contract. Most PC software is sold with a lifetime license. In some cases, the license may restrict the software to use only on a single machine or processor, that is, the lifetime may refer to the lifetime of the machine not to the lifetime of the client or the software.
- Fixed period license. Fixed period licenses permit use of the software for a fixed period of time, usually one year. Continued use of the software may require the payment of an annual license fee. Many mainframe, network, and

server software products come with fixed period licenses. Some vendors insert “time-bomb” components in their products to disable the software after a certain date. The presence and nature of the bomb may constitute an important evaluation feature. In the most dangerous versions, the bomb fully disables the product without warning on the expiration date. Instant disabling can electronically shut down the product and potentially damage a company that wants to withhold payment from a vendor over a legitimate dispute.

Costs and Prices

Because of the wide range of acquisition options and the varying content of services provided, comparing the cost of alternative products may require careful analysis. Some systems are sold for a one-time charge, some are leased, and others are sold with an optional or required annual fee for support and upgrades. Some systems are licensed by site, some by user, and others permit network use or offer a company license. Some include upgrades forever or for a defined time period; others do not. Some include allowances for installation and/or for maintenance.

Contracts to develop and/or install software, perform maintenance, or provide other services may involve one of the following costing schemes:

- **Fixed price.** The client pays a predetermined price for the service. The vendor benefits from spending as little as possible to provide the specified service.
- **Cost plus fee.** The client pays the vendors direct costs for labor, materials, and other expenses plus a percentage or set amount for profit. The vendor may see little incentive or even a disincentive to control costs.
- **Incentive.** If the project costs less than expected, the vendor and client share the savings. The vendor and the client benefit jointly from spending as little as possible to provide the specified service; however, the vendor may prefer to keep employees and equipment busy more than to receive a share of reduced costs.

Many combinations and permutations of cost or price schemes are possible. Each approach has both advantages and disadvantages. The team must list all of the relevant features to make a meaningful comparison. Even with the best analysis, uncertainty remains. The team and client may never know in advance the answers to such questions as how much maintenance will cost or how important upgrades are. In addition, the monies paid to the vendor may represent only a small fraction of the total life cycle cost for the product. Such additional costs as training, staff, related hardware and software costs, and installation costs can amount to far more than purchase price. The team may wish to refer to Chapter 9 to prepare a complete cost analysis.

No organization likes to overpay for a product or service, and most organizations consider cost management to be a survival skill. Despite this background, most managers rank cost reduction well down in the list of reasons for outsourcing. In short, cost is only one of the concerns in a successful sourcing selection. Most organizations recognize that a low-cost vendor may not deliver the most cost-effective solution. Joe’s TV shop may pull Ethernet cable at a very attractive price, for example, but if the connections are loose or the strains too

large on the cables, the network may perform unreliably. Strict supervision of unqualified vendors seldom yields a good outcome.

Laws and Regulations

Organizations operate under a number of governmental laws, rules, and regulations. These rules include such areas as accounting practices, record-keeping requirements, and privacy protection. Products can create liability for the client. An expert system that purports to provide legal advice could open the client organization up to civil liability. An accounting system that fails to track customer assets correctly likewise could involve the client in expensive litigation.

Laws and regulations may vary by country, state, and organizational activity. For example, energy companies, airlines, and hospitals operate under quite different legal and regulatory rules. A product vendor may not know or understand the rules that apply to a specific purchaser. Legal and regulatory issues that apply to the client's organization normally lead to mandatory features. The team has an obligation to raise legal and regulatory issues explicitly with the client, to identify the features that apply to the client, and to evaluate them as a part of the sourcing process.

Product Flexibility

Many products, especially larger ones, will not meet exactly the client's required features. Vendors try to deal with this issue with **product flexibility** options that allow the client to customize the product. The client selects the options that come closest to the features the client wants. A wide range of options facilitate tailoring a product to the client's needs and also may accommodate future changes in requirements. However, options may increase the product's costs for purchase and support. Installing options takes time and effort, and some products may require more effort than others. In a large ERP product, setting all the available options and installing the system can require several years of work by a large team. A client who does not need a multitude of options may find a simpler product more cost effective.

Even with options, a product may not satisfy a client. Often clients want the IT organization to "write some code" to modify the core functionality of the purchased system. The golden rule of buying products is "Change the requirements in place of changing the product." Most organization's current operations result more from historical accidents than from excellent design. The package's functions may represent more careful and insightful thought about good approaches to the problem than the current operation. Users and clients tend to resist changing their requirements on first contact but after discussion and thought often agree that changing to fit the package offers advantages.

If unfilled client requirements remain after serious thought and discussion, a cost-effective way to add features to a purchased product involves writing custom "wrapper" modules that interface with the purchased system through application program interfaces, (APIs) allow the product to export data to and import data from other programs. When the client desires modification, the presence and functionality of APIs can represent an important feature for product evaluation. The

team also should determine the availability of experienced staff or consultants in the market area to help with customization problems.

The least desirable and most dangerous alternative to adding functionality involves modifying the program code of the product. Code modification generally leads to endless problems and great client dissatisfaction with both the vendor and the IT group. Some vendors refuse to share the source code of a product with their customers, making modification of the code very difficult and expensive. In addition, modifying the source code usually voids any warranty or service guarantee.

Even worse, modifications to source code make installation of upgrades and patches costly or infeasible. Every time the client installs an upgrade or patch from the vendor, the IT group always must analyze and debug the effect on the modifications and may need to completely recode the custom modifications at a cost to IT that may exceed the cost of the original modification. Vendors may issue upgrades and patches several times a year or more often. Clients who fail to install patches and upgrades may lose significant pieces of functionality, or the product may cease to function at all. The team should strongly caution the client against any modifications to the source code for a package.

Contract Compliance

Contract compliance issues refer to exercising and enforcing the terms of the outsourcing agreement or contract. With any agreement, the organization must identify and track appropriate performance metrics. Should the organization discover that it is not getting the agreed-upon performance, the organization may decide to pursue a remedy. The kind of remedy may depend somewhat on the company's relationship with the vendor. With a *commodity relationship*, the purchase and warranty agreement should spell out exactly what performance is expected and what penalties apply for nonperformance. The scope and effectiveness of the agreement may become part of the features analysis.

Typically a client cannot just buy software outright; instead the vendor gives the client a license to use the software with a number of different license restrictions. License provisions and restrictions may vary from one vendor to another in ways that impact the client's intended uses for the product. If so, the contract features become part of the features analysis and comparison. Contract compliance involves serious issues. The Software and Information Industry Association (SIIA), through its Software Publishers Association (SPA) Anti-Piracy Division, actively pursues the identification and prosecution of organizations and individuals who illegally use software.

Some off-the-shelf products come with standard warranties. Larger, custom products and services are protected by service agreements and performance penalties. To minimize expensive legal wrangling, the contract must state clearly and explicitly all the agreements. Agreements should specify penalties for nonperformance. If a vendor promises to deliver a product by May 1 or specifies that a product will support 10,000 transactions per day, and the terms of the agreement are breached, the contract should indicate the specific penalties or consequences for nonperformance. Although the client can put large penalties into contracts, the costs to the client for nonperformance often exceed the penalties. The main purpose of a penalty is to provide a focus and incentive for the

vendor to comply with the terms of the agreement. Vendors who fear that they may incur a performance penalty may increase their bid price to cover the risk.

Warranties and performance agreements, however well written, still require careful monitoring and enforcement. If a person buys a defective toaster from a retail store, he or she can return the defective toaster to the store or vendor for a refund or exchange. Guarantees on expensive products or service agreements, however, lead not to automatic replacement but to the remedies specified in the purchase contract or to complex, expensive negotiation or litigation. In addition, once a product or service is acquired, the organization needs to assure that it continues to meet its ongoing needs. With large products, maintenance issues may cost from 2 percent to 10 percent of the purchase price of the product every year. The client needs to check that all agreed updates and patches are sent, any identified bugs or performance glitches are resolved satisfactorily, and all service level standards are met.

For an association with the vendor for a large package or service contract, the formal agreements should focus on dispute resolution. The agreement should anticipate that problems might come up and specify a process for settling them. Association sourcing arrangements work best when the client assigns a person to work on a continuing basis with people from the vendor organization to solve problems and promote a positive working relationship.

In the best of circumstances, the time may come when the client or vendor wishes to terminate the association contract. Especially with large contracts, the termination features belong in the features analysis. With *termination-for-cause*, the aggrieved party must demonstrate that sufficient cause exists to void the agreement. Termination for cause often leads to legal action over whether the cause was sufficient under the contract terms. *Termination-at-will* specifies that either of the parties may end the agreement at their sole discretion, and the contract specifies what should happen at the time of termination. The agreement should include notification requirements, the penalties, fees or settlements that apply, a specification of data ownership, and any rights to service or product use during a transition period.

Organizational Fit

Some products fit a particular organization better than others. One product may support manufacturing; another may work with marketing. Sometimes products contain specialized features for an industry, for example, insurance or a medical practice. Choosing a product written for an organization similar to the client's may provide a product designed and customized to provide the most common functions for such organizations. A general product or one developed for another line of applications may work but probably will require more customization than one developed specifically for similar applications.

VENDOR ROLES

When a client purchases a product or service, the client selects not only the product but also the vendor. With the association model used for procurement of mid- and large-sized packages and for most consulting and ASP contracts, the

characteristics of the vendor may weigh as much or more heavily on the decision than product features. A good product from a poor or unstable vendor can lead to serious problems for the client.

Vendor Features

The team should investigate and evaluate the important **vendor features** and characteristics. Some indicators of vendor desirability include **vendor stability**, delivery record, and support record.

Vendor Stability

When possible, clients want to contract with vendors who will continue in business over the life cycle of the product, anywhere from several to 20 or more years. Many IT vendors start up, sell a number of products or services, and then go out of the IT business. When the vendor goes out of business, the client is left with the expense of supporting the product or replacing the product. The client may find that self-support of some products is difficult or even impossible.

The length of time the vendor has operated in the market is one indicator of stability. A company that has operated for five or more years in the market of interest to the team offers a good chance for a stable long-term association. The company's market share for the product type that is being considered offers another indicator of stability. A company with one of the larger shares in the market tends to have the resources and incentives to support the product and remain in the market. Small divisions of large, stable companies may not offer much stability unless the division meets longevity and market share criteria on its own. Unless the small division can or does meet profit, ROI, and market share targets, the parent company may sell or close the small division.

Vendor Support

Other business issues that impact the likelihood that the company can provide ongoing service include a local office, a strong balance sheet, good profits, patent or copyright protection for products, a core group of talented employees, and related issues. Clients may depend heavily upon support for the product provided by the vendor. The vendor may provide support to tailor the product to the client, install the product, maintain the product, fix any design problems, train staff or users, and provide improvements. The team may wish to examine such factors as:

- The record and reputation of the vendor for support.
- The availability of technical support personnel.
- The conditions or rules the vendor sets for the client to obtain vendor support for problems. For example, vendors may provide support 24 × 7 or only during regular working hours. Vendors may act to fix all the problems or only the problems they agree are "covered."

- The costs for the various types and levels of vendor support. Some types may come free with the product or at preset rates; other may cost whatever the vendor decides at the time they are needed.
- The type and level of help desk service the vendor provides. A good help desk can allow the client to resolve many problems quickly without waiting for a service call.
- The expected frequency of upgrade releases and the expense and skill required to acquire and install them. Once a client has identified, acquired, and installed a product, the organization still must assure that the product functions effectively on an ongoing basis. For many products, an effective ongoing operation requires installing patches and upgrades.

Delivery Record

The company's record for delivering products on time that fully meet the specifications is an important vendor feature. Vendors that respond promptly and truthfully to queries and concerns are more likely to respond as agreed and on time with products. The failure of a vendor to deliver products as agreed can devastate a project and cause the client to incur large expenses.

Product/Vendor Selection Issues

Finding the information needed to make a good product and vendor selection decision can pose difficulties. Sales staff, brochures, and sales representatives tend to present every product as outstanding and the vendor as stable, innovative, and dedicated to customer support. In short, vendor literature and representatives tend to present every vendor and product as the best or certainly far above average. Clearly, the team should attempt to find an independent confirmation of vendor and product performance. Published reviews and consultant reports provide a good starting point. A number of IT and business magazines conduct periodic product evaluations that often include reports on product support. These reports can include responses from a wide variety of respondents. However, the product the team is considering may not have any published reviews, the reviews may not cover the issues of most importance to the team, and some reviews are inaccurate and misleading.

Consulting and specialized evaluation organizations conduct formal product reviews, either by request or for periodic distribution to clients. Consultants bring a wider range of experiences to the analysis than any one organization possesses. They may have conducted tests, and they often provide valuable insight. Consultants provide information for their livelihood; most charge fees, sometimes very high fees, for information. Specialized consulting organizations focus largely or exclusively on product and vendor evaluation and on providing guidance on IT issues. These organizations tend to provide high-quality evaluations and also point out functionality, performance, interoperability, and other issues. However, consulting organizations that provide information may not know the specific application or situation faced by the client unless the client contracts with the organization for a customized evaluation.

Previous customers offer a helpful source of information on both vendors and products. Most vendors will provide a list of “satisfied” customers. While the list may contain bias by omitting customers who have complained often and/or stopped using the product or vendor, the customers on the list can relate some specific positive or negative experiences that can help the team. Industry trade associations and satisfied customers may identify other customers that the team can contact for a broader sample of views. Other customers’ experiences, while helpful, may not fully define what the client will experience. The product may work better or worse in the client’s application depending on the use patterns and expectations, and the vendor may provide better or worse support depending on the people assigned, location, and other factors. Project teams can ask the client for permission to contact other firms and vendors in the name of the client’s organization. Vendors tend to respond better to a contact in the name of a person with the funds or authority to buy the product.

Looking at the aforementioned vendor criteria can help the team and client to select a satisfactory product/vendor combination, but will not guarantee a successful association. In the IT market, a large number of vendors with a long history of success, for example, DEC, Compact, General Electric, and RCA encountered problems and either withdrew from IT markets or merged. Sometimes another organization takes over the support of the products offered by the disappearing vendor, but many times some or all of the products become orphans without any support or upgrades. A client with hardware or software products from a vendor that has left the marketplace may encounter large expenses and disruptions in a forced move to an alternative product.

The team may decide to select a vendor that does not meet the above conditions for many reasons. For example, the vendor may be the sole or a local provider of the desired product, may offer an attractive financial deal, may possess previous experience, and so forth. However, the team should realize and inform the client that dealing with the vendor may involve additional risk.

REQUEST FOR PROPOSAL

When the procurement of product or service costs involves a significant amount of money, most organizations require and/or use a competitive bidding process. Federal, state, or local laws generally require public organizations to use competitive bidding. Many organizations initiate the information-generating or bidding process by preparing a document resembling one of the following:

- Request for information (RFI). A document sent to vendors describing a problem or need and asking for information about products or services to address the problem. The information received is used to refine specifications and/or identify potential bidders.
- Request for quote (RFQ). A document sent to prospective bidders with specifications for a commodity-type product or service and asking for prices and related conditions.

- Request for proposal (RFP). A document sent to prospective bidders with background and specifications for an association-type product or service and asking for a detailed proposal on how the bidder would work with the client to meet the needs.

Many information system outsourcing projects fall in the RFP category. The client wants to establish a relationship with the vendor to acquire a software system or another product or service. The RFP contains the information that a vendor needs to submit a bid—information similar to the design specifications for a proposed system plus procedural information on how, what, where, and when to bid. The organization makes the same RFP document available to all the potential bidders.

The goals of the RFP process are to encourage vendors to (1) perform part of the design work for a solution and to (2) compete for the contract by prepared comprehensive bids that respond effectively to the organization's problem. The explicit specifications help the bidders to prepare a bid that addresses the organization's problem. The fair and open competition encourages bidders to spend the time and effort needed to prepare a responsive bid. Vendors do not want to incur the costs to prepare a bid if they have little or no chance of winning the competition.

Some RFPs include evaluation criteria in the bid. Not all organizations select the lowest bidder, but all organizations should follow the rules and criteria they provide in the RFP. Organizations, especially government offices, that fail to follow the procedures outlined in the RFP run the risk of lawsuits. At a minimum, an organization that regularly violates the conditions of or misuses the RFP process will find few qualified bidders for their projects.

RFP Content

Most larger organizations have their own format for an RFP, and these formats vary greatly. Some organizations may have multiple formats to use in bids for different classes of products. The RFP structure for vendors to bid on such commoditylike items as PCs may look quite different from one used to select a vendor to build and maintain a custom system. A typical RFP might include the following sections. Each of the section headings in the outline is explained in the sample GB Video RFP in the next section.

- Instructions
 - Objectives
 - Contacts
 - Timetable
 - Bid and Contract Requirements
 - Delivery
 - Validation
 - Review Points
 - Termination
 - Costs and Payments

- Subcontractors
- Warranty
- Solution Requirements
 - Background and Problem Statement
 - Solution Constraints
 - Current Operations
 - Problem Analysis
 - System Specifications
 - Mandatory Features
 - Desirable Features
- Bid Contents
 - Work Plan
 - Performance Record
 - Schedule
 - Resources
 - Costs
- Evaluation Method (sometimes not given to bidders)

Vendor selection often follows a multistage process. The selection team may select the set of vendors to receive the RFP or may make the RFP available to all interested parties. Each vendor can respond with a **bid** that defines the product the vendor will provide, the price and the terms. Once bids are received, the team often screens the responses to classify bidders as qualified and nonqualified. The qualified bids meet all of the minimum requirements in the RFP. The nonqualified bids are set aside. The best qualified bids go through an in-depth evaluation to select the winner. Sometimes the team offers a group of vendors with “good” bids a chance to make a “best and final” offer, whereby they can revise and improve their original proposal prior to the final selection. Some organizations will select a winning vendor and then negotiate with the vendor to arrive at a final agreement. During the negotiations, the team may ask the vendor to suggest ways to improve the product and/or to reduce the cost.

GB Video RFP Example

Figure 10.1 shows an example of an RFP prepared for vendors or contractors who might bid on the contract development of a GB Video Rental System. The Instructions section, comprised mostly of material prepared specifically for the RFP, gives procedural requirements for submitting a bid. The bid and contract procedures in the RFP in Figure 10.1 are only illustrations of possible content. Practices and legal requirements will vary from organization to organization and state to state. The analysis and design work described in the Project Definition and Proposed System sections of this book should provide much of the content for the Solution Requirements section of the RFP. The Evaluation Method section is used for the GB Video Bid Analysis example in the next section.

FIGURE 10.1 RFP for GB Video

G.B. Video, Inc.
Request for Proposal: Computerized Video Rental System

INSTRUCTIONS

GB Video, Inc. is requesting bids from experienced systems developers to design and build a new computer-based system for the rental and return of videos because the current manual system maintains inadequate records and is too slow and expensive. Each bidder should carefully review this RFP and follow the guidance therein. Bidders who fail to follow the guidance in the RFP including due dates for meetings and materials may be summarily eliminated from further consideration.

All parties to the process are expected to behave with good faith and to resolve any differences amicably and fairly within the intent of the RFP and any resulting contract. Unresolved disputes, if any, that may arise from any part of the bidding or from a resulting contract will be adjudicated under the laws of the state of Oklahoma.

Objective

The new system should contribute to improved customer service and lower handling costs for each rental and return transaction. GB Video intends to have the new system that meets the specifications set forth in this RFP installed and running in all stores by April 1 of next year.

Contacts

All contacts regarding and materials submitted for this RFP should be addressed to:

Judy Olijer, Director of Purchasing
GB Video
100 Data Street
Jackson, OK 73099
Phone (405) 555-0011 FAX (405) 555-0022

e-mail: Judy.Olijer@gbvideo.com

GB Video takes no responsibility for and is not bound by communications with any other GB employee or any other person.

Timetable

GB Video plans to follow the procurement timetable shown below. GB Video reserves the right to extend the dates if circumstances warrant. GB will provide as much notice as possible of any changes to the affected bidders.

- **May 4, 3:00 p.m. Pre-bid Conference.** A mandatory pre-bid conference for all interested bidders will be scheduled for 3:00 p.m., May 4, at the home office of GB Video, 100 Data Street, Jackson, OK. Only bidders present at this conference may submit bids.

- **June 4, 5:00 p.m. Bid Due Date.** Final written bids must be received at the GB Video home office (the contact address above) by 5:00 p.m. on June 4. FAX and e-mail delivery are not acceptable.
- **June 24, 5:00 p.m. Notification.** Finalists will be notified by 5:00 p.m. June 24 of their status. A best and final offer must be received by June 30.
- **July. Presentation.** Finalists may make a final presentation during the month of July.
- **August 15. Award.** GB intends to award the contract by August 15 with a required completion date of April 1 of the following year.

Bid and Contract Requirements

GB Video plans to incorporate the following general requirements into the contract. GB reserves the right to negotiate the exact terms with the winning bidder.

Delivery. The vendor should complete, gain acceptance of, and install the system by April 1. Bidders must submit a proposed solution that bidder can reasonably accomplish by the delivery date. A penalty of 2% of the total contract price shall be assessed for each week that completion is delayed.

Validation. Upon written acceptance of the production system, GB Video has 90 days to identify any previously undetected system deficiencies. A deficiency is incorrect operation or lack of any feature contained in or reasonably implied by the bid. GB will notify bidder in writing of deficiencies. Bidder agrees to correct deficiencies within 30 days of notice at no cost to GB Video. If vendor fails to correct deficiencies, GB may reduce the payment due the vendor in the amount of the fair and reasonable cost for another contractor to correct the deficiencies.

Review Points. The project work plan should identify and clearly define three milestones for GB Video review and authorization to continue.

1. Final Design Acceptance.
2. Functional and Operational Approval of the Production System.
3. Final Testing and Installation.

Termination. GB reserves the right to terminate at will the contract with the bidder at any of the above review points.

Cost and Payment. The bidder shall specify a fixed, lump-sum price for the total project and costs for termination of the contract at each of the above review points. GB will pay all sums due in full at and only at the time GB accepts the software or at the time of termination of the contract.

Ownership Rights. The vendor will give GB Video a lifetime license for an unrestricted number of GB stores for the rental and return software produced by the

vendor. GB agrees not to sell or lease the software to third parties. GB shall receive the source code and have the right to modify the software, but modifications made by any party other than the vendor may nullify the warranty.

Subcontractors. The bidder will indicate in detail any parts of the project, for example coding, that the bidder plans to subcontract and fully identify the subcontractor. Any subcontractors or subcontracted work not in the bid require prior approval by GB Video.

Warranty. Should the system cease to operate as observed at the time of final validation through no fault of GB and its agents, bidder agrees to restore the system to proper operation at no cost to GB for 5 years from the date of acceptance. Vendor agrees to perform modifications and updates as requested by GB Video using the cost reimbursement structure for such work as specified in the bid.

SOLUTION REQUIREMENTS

The materials in the following section provide background and specifications for the video rental system.

Background and Problem Statement

GB Video operates three video stores in towns of about 10,000 people. Each store operates largely independently although the headquarters performs some functions for all three stores, for example, payroll, purchasing, and accounting. The company employs 37 people and realized revenues of \$1.5 million and profits of \$133,000 last year. The new system is mission-critical for GB. The company plans to open additional stores if the new system results in improvements. The new information system should increase profits by reducing labor cost per transaction and eliminating the cost of videos rented to nonmembers and not returned. Faster checkout service and better selections can lead to increased customer satisfaction, more members, and higher sales. GB looks at labor costs, profits, and revenues per store as major performance measures and at the total number of active members and the number of rentals per member.

Solution Constraints. GB will consider solutions that will realize a payback in two years or less and cost less than a total of \$200,000 to acquire and implement. GB is willing to consider changes in function and organization if the changes provide significant benefits to GB. Adequate air-conditioned space in the headquarters exists for a server and network room. The vendor may bid to supply equipment or specify with the option for GB to acquire the necessary equipment. Equipment costs are included in the \$200,000 limit.

Current Operations. GB currently operates in a manner similar to most video stores. However, the current GB system is manual and uses paper forms and files. Appendix A contains a description of current operations.

Problem Analysis. GB has identified the following problems in the current situation:

1. Because of the manual system, GB requires more clerks than similar stores with automated systems, resulting in a higher cost and longer elapsed time per transaction.
2. Customers complain about long lines and slow checkout of rentals. Some frustrated customers dump their videos on the counter and go off without completing the rental.
3. Delays occur when several clerks wish to use the member or video card file at the same time.
4. Mistakes are common. For example, the video card may indicate that the video is on the shelf—it shows a return date and no new rental date—but the actual video cannot be found. Clerks make mistakes in computing the charges. Customers often correct overcharges but remain silent when the clerk undercharges.
5. People, including some members, use false member numbers and names to rent videos. If the customer does not show a member card, the clerk is supposed to check the member file. Because of the time and effort needed to check the file, the clerks omit checks most of the time, especially when the store is busy. GB experiences a higher nonreturn or loss rate for videos than other similar stores.
6. Accounting gets busy and does not send overdue notices to customers on a timely basis. Sometimes a video is several weeks overdue before the customer receives a notice. Customers use the late notice as a reason to refuse to pay overdue charges.
7. Because of the expense of preparing a manual mailing, GB does not do any direct mail marketing to members. Other stores have successfully used direct mail marketing to increase revenues.
8. The rental data summarized by video is costly to prepare and of questionable accuracy. No data exist on customer preferences. Purchasing often uses guesses, estimates, and periodic direct observation to determine which videos and how many copies to buy.

GB video believes that installing a computerized rental system that meets the specifications in the RFP should address these problems.

System Specifications

The materials below describe both mandatory and desired features for the proposed system. The bid is expected to include all of the mandatory features. The bidder must clearly identify any mandatory feature not included. The bidder should note the desirable features included in the bid plus the extra cost, if any, for them.

Mandatory Features. The proposed rental and return system will include the following mandatory functions:

1. Membership
 - a. Collect data and store data on new members.
 - b. Update data for existing members.
 - c. Issue a member card to members.

2. Rental
 - a. Rent tapes only to members.
 - b. Create and store a rental record with identification of member and video.
 - c. Adjust an inventory record to reflect the rental.
 - d. Issue a receipt to the customer/member.
3. Return
 - a. Update the rental record and the inventory record to reflect the return.
 - b. Calculate the overdue charge, if any.
4. Overdue
 - a. Create overdue notices for videos that are overdue.
 - b. Bill customers for rentals more than 14 days overdue.

Detailed specifications for the proposed system's mandatory features appear in the appendixes listed below.

- Appendix B. Narrative Specifications
- Appendix C. Proposed Data Flow Diagram
- Appendix D. Proposed Entity Relationship Diagram
- Appendix E. System Metadata

Desirable Features. The bidder should note which if any of the desirable features are included in the bid and the extra cost, if any. Desirable features in approximate priority order include:

- Expandable at no or small cost for up to 100 additional stores and sites.
- Contain or compatible with a Web-enabled rental function.
- Operate in a technology environment for which experienced people are available in the Oklahoma market.
- Generate rental number and customer number for a new member in place of scanning or other entry method that requires clerk intervention.
- Automatically calculate sales tax for multiple city and county locations.
- Generate ready-to-mail overdue notices with postage.
- Contain the option for e-mail overdue notices and e-mail advertising or information messages to customers.
- Generate internal reports and communications at specified times and on-demand.
- Minimize manual handling of paper documents.
- Intuitive and user friendly.
- Compatible with the current organizational environment at GB Video.

BID CONTENTS

Each bid must supply at least the following information:

1. **Work plan.** The statement of work for the bid must: (a) demonstrate that the contractor understands the overall project and goals; and (b) indicate in detailed fashion how the contractor will proceed to work to meet the specifications and requirements; and (c) identify clearly any proposed deviations from specifications and requirements.

2. **Performance record.** Evidence of successful performance on similar projects. Bidder will provide a project summary with a list of references for verification of experience in similar projects or a suitable alternative.
3. **Schedule.** The date when each part will be complete and a date for the completion of the total project.
4. **Resources.** Evidence that bidder has the proper resources and capabilities to generate the desired results and specify exactly which resources will be dedicated to this project. Include people, skills, and other resources. Preference will be given to vendors with a local office.
5. **Costs.** An enumeration and explanation of relevant costs and charges for the project. Although cost is important, the GB team will evaluate bids on the best total value to GB, not necessarily the lowest cost.

EVALUATION METHOD

A panel of individuals from GB Video will evaluate the bids based on material in the bid and supplementary materials, for example, talking with customers, collected by the GB team. The items in the evaluation and their point weights are described below.

1. Compliance with requirements and specifications: 30 points
The degree to which the submitted bid conforms to or exceeds the specifications in the RFP. The work plan should demonstrate in detailed fashion that the contractor understands the overall project and goals and will meet the mandatory specifications. To receive the highest score the contractor also must commit to some or all of the desired features.
2. Experience with similar projects in other organizations: 30 points
The extent to which the bidder provides evidence of successful performance on similar projects. A summary of similar projects with a list of references to contact for each is the preferred way of demonstrating this experience.
3. Ability to meet required deadlines: 10 points
The level of confidence that the contractor will satisfactorily accomplish the required tasks within the specified time frame. Demonstrated ability to meet deadlines on other jobs is one indicator. A demonstrated record of completing work prior to the deadlines will receive additional points.
4. Organizational capacity: 20 points
The extent to which the bidder provides evidence that bidder has the resources and capabilities to generate the desired results. This measure includes both human and technical resources. An outline of the resources the vendor will dedicate to this project will clarify the level of available capacity.
5. Costs of the work: 10 points
The total cost and the cost for each of the three parts of the RFP.
6. Supplemental features. The evaluation team should note any features of the bid not reflected in the above measures.

APPENDIXES

- Appendix A. Current Operations. (See Chapter 7.)
- Appendix B. Narrative Specifications. (See Chapter 8.)
- Appendix C. Proposed Data Flow Diagram. (See Chapter 8.)
- Appendix D. Proposed Entity Relationship Diagram. (See Chapter 8.)
- Appendix E. System Metadata. (See Chapter 8.)

FEATURES EVALUATION

Once the team and client identify the system features of interest for a procurement, the team must formulate a process for evaluation of potential vendors or contractors. Normally, the team prepares a features matrix or table to summarize and compare the results. The matrix lists the features on one axis and the alternatives on the other. Each feature/alternative box in the matrix is assigned a measure of merit, that is, a measure that describes how well the alternative achieves the feature. Table 10.2 shows the format for a features analysis matrix. The matrix may list anywhere from several to dozens of features. Each box in the table gives the ranking for the solution/feature pairs. Ranking methods are described in the next section.

Ranking Methods

Three commonly used ways to assign measures of merit are a features list, economic value, and rating points. The client should determine the preferred approach. A **features list** rates the features of a product using the format shown in Table 10.2. Many published product reviews use this approach. A number or symbol rates each feature of each solution. The table may or may not show a sum of values for each solution. The client selects the solution with the profile the client considers to be the most desirable. The ratings list approach is commonly used in purchasing lower cost products, for example, printers, scanners, and displays.

With an **economic value analysis**, the client assigns a dollar value of each solution/feature box using the format in Table 10.2. In some cases, the team can estimate the values. In others, the team may ask the client to specify the value,

TABLE 10.2
Features
Analysis
Matrix

Features	Solution 1	Solution 2
Feature 1	10	6
Feature 2	5	7
Total	15	13

for example, “A Web-enabled system is worth an additional \$7,000 dollars to me.” The costs for the solution, which include purchase, installation, and operations, appear as negative values. The overall score of each solution is the sum of the values of each feature. The dollar value approach tends to consume a lot of interaction time with the client. The field project team may lack enough knowledge of the client’s organization to estimate the dollar value of a feature or function to the organization. As discussed in Chapter 9, estimating costs and benefits may involve great difficulty. A poorly executed dollar value analysis is less convincing than none at all.

Many organizations prefer to use a **rating point** scheme. Sometimes mandatory features are assigned only a Yes or No value. The features matrix starts with a list of mandatory functions and features. Each solution alternative receives either a Yes or No for each feature. One or more No values for a solution alternative results in rejection of the alternative or a review with the client to determine a possible change in requirements. For all the other such features as costs, vendor features, and desirable features, the client selects or agrees to a weighting for each one.

The weights reflect the priorities or values of the client or the client’s organization. The weights should be consistent with the strategic alignment analysis for the project. A feature that the client values highly receives a high weight. The client may choose to assign point weights to each feature or to use a percent weight. The percentages, when used, should add up to 100 (or 1 when the client converts percentages to fractions) over all the features. For example, the client may decide that the solution cost should have a weight of 50 percent (0.5) in the matrix. The team then assigns points (perhaps with a 0 through 10 scale) to measure how well each product satisfies the feature. With a 0 through 10 scale, the lowest cost solution may receive a 10 and the other solutions a lower number. An example of a weighted features analysis appears in Table 10.3.

The actual weighted score for that feature is the product of the weight and the rating score. Each solution score is the sum of the individual feature scores. The advantage of this system is that the team does not have to understand the organization as intimately to execute the evaluation. The client determines organizational fit by using the percentage weighting. The team can then rate each feature relatively independently of the organization priorities. As shown in the example, the solution with the best total score for the ranking by themselves may not be the same as the solution with the best total score for weighted rankings.

TABLE 10.3
Weighted
Features
Analysis

Features	Weight	Solution 1		Solution 2	
		Ranking	Weighted	Ranking	Weighted
Feature 1	0.7	6	4.2	9	6.3
Feature 2	0.3	10	3.0	5	1.5
Total		16	7.2	14	7.8

Identifying Candidate Solutions

The client may ask the team to identify solution candidates. The first place to look is vendor sites on the Web (most IT products have a Web site). A second place to look is Web sites for trade associations and special interest sites. Many clients have some suggestions for solutions and have either a library or subscriptions to trade journals that they will share. The client may also belong to one or more trade associations or may refer you to a contracted consultant or research group. Vendors, when contacted, may offer additional suggestions for solutions. Instead of asking the team to identify solutions, the client may ask the team to prepare a plan and specifications that will allow the client to search for solutions.

Assigning Ratings

Features analysis approaches require someone to assign values. The team should strive for objective and informed ratings. When team members are not experts in the technology or system, the team may wish to consult more experienced people. In government organizations, objectivity and consistency often are legal requirements. Fair, objective evaluations can keep vendors coming back to work with an organization.

Any careful evaluation of all possible alternative solutions may overwhelm the team. A good first step is to eliminate solutions that do not meet the mandatory requirements or exceed the client's budget. Vendor information in brochures or on the Web may provide enough information for a first pass. The team should ask vendors for a contact person, brochures, and other documentation as early as possible in the process. The vendor contact person often will clarify issues by telephone. The objective of the first analysis is to reduce to a handful the number of solutions for a more thorough review.

As noted earlier, customers, trade and professional journals, and consultants may have ratings or information relevant to ratings for one or more of the solutions. If the team cannot find a published review, then the team must set up a process for scoring the products. A reasonable approach is to have two or more evaluators (either individuals or groups of team members) conduct independent reviews of the features of the products. The team can discuss disagreements and come to a consensus. The groups make their evaluations based on as much information as they can find.

The team also might install copies of each of the products and have a team of analysts and users try them out and assign ratings for the matrix. Most field projects are small enough that installation and use of several products is feasible. The team can apply the test procedures discussed in Chapter 13 for part of the comparison. This approach may require the client to provide permission for the team to use the organization's name in contacting vendors. "Company X is considering purchasing your product," usually gets much better response than "A student team wants a copy of your product for a class project." This approach may require the team to obtain test data—a possible sensitive issue for data about customers or employees. The client may require the team to sign a confidentiality or nondisclosure agreement before accessing data.

FIGURE 10.2 GB Video Bidder Analysis

As instructed by Mr. Cosier, the team prepared an RFP and sent it to six potential bidders that the team identified. Four bidders submitted proposals. Two of the four bids were deemed nonqualified and were eliminated from the analysis. The team prepared the bid summary below for the two finalists.

Bid Responses	Adv. Software	OkieComp
Met or exceeded minimum qualifications	Yes	Yes
Meet all mandatory and desired specs	Yes	Yes
Initial software cost	\$109,950	\$94,500
Cost to load data into the new system	\$4,700 (with a subcontractor)	\$5,000
Cost for upgrades and modifications	Labor + 80%	Labor + 70%
Experience with similar projects	Good	Very good
Customer responses on delivery performance	Very good	Excellent
Resources available for the project	Adequate	Excellent

An evaluation committee of two team members, the RFP consultant, President Cosier, and Purchasing Director Olijer read the bids and assigned rankings to the evaluation criteria in the RFP. The resulting evaluation by the committee appears below.

Features	Weight	Advanced Software		OkieComp, Inc.	
		Ranking	Weighted	Ranking	Weighted
Specifications	0.3	10	3.0	10	3.0
Experience	0.3	8	2.4	9	2.7
Deadlines	0.1	9	0.9	10	1.0
Capacity	0.2	8	1.6	10	2.0
Cost	0.1	7	0.7	10	1.0
Total	1.0	42	8.6	49	9.7

The evaluation committee recommends that GB enter into a contract with OkieComp, Inc. Both vendors are well qualified and submitted good bids. Although the differences are relatively modest, OkieComp received the highest raw and weighted scores and is the vendor preferred by Ms. Olijer. Conversations with the proposed project manager from OkieComp convinced the team that the company understands well both GB Video and the video rental industry. While both bidders have been in business for more than 10 years, the high failure rate of software firms poses some risk for future support from both bidders; the committee could not identify any risk differences between the firms. If for some reason, GB is unable to reach a contract agreement with OkieComp, the committee finds the Advanced Software bid an acceptable alternative.

Some organizations use variations of this approach for some expensive purchases. However, installing and running a number of software systems can involve large expenses and a lot of time. For a large system, converting data, customizing the product, and training users can take months or years. A simulation or benchmark test may offer a better alternative. A simulation can calculate performance based on published properties of the system or can run the system against a generated set of test data. A benchmark uses an existing installed version of the package and a sample of actual data to measure the performance of the system.

Outcomes

The client may ask the team to provide (1) the evaluation framework; (2) the results of the evaluation; and/or (3) a recommendation. When asked to recommend a specific product, the team, as discussed in Chapter 9, can review the critical features of the recommended solution, organize the features in a clear way, and discuss the rationale for the recommendations. The team's recommendation should consider all of the concerns raised by the client and not just look for the "best" score. For example, the team might say, "Since these products received similar total scores, the team recommends X because it appears to fit best with the strategic alignment of the project."

GB Video Example of a Weighted Features Analysis

Figure 10.2 shows an example of a weighted features analysis for GB Video based on the valuation criteria identified in the GB Video RFP in Figure 10.1. The RFP specifies point or percent value weights for each feature. In the table the team converted the percents to fraction weights.

Contracts

After the bids are opened and the winner selected, the organization needs to execute a contract for the outsourced product or service. Many of the conditions for the contract may appear in the RFP. Most organizations have rules, standards, or procedures for contracts and also work with a legal advisor who will wish to be closely involved with any contract for a significant amount. Sometimes the organization has the legal and procedural freedom to sit down with the winning bidder and negotiate the terms of the deal. Such negotiations can benefit both parties. The vendor often can suggest changes that reduce cost or delivery time with little reduction in benefits for the client. The vendor also may suggest changes to improve function or performance at little or no additional cost. Most government entities prohibit such negotiations, but many nongovernment organizations find them helpful.

Summary

The paths of information system solutions resemble a tree. During project definition, most teams follow a similar path, the trunk of the tree. As work proceeds, the nature and requirements of different projects take teams along diverging

branches. During the evaluation process discussed in Chapter 9, the team selects a sourcing option—to build or buy. As a result, teams may follow quite different paths in the system delivery stage. For example, some teams will build or develop detailed specifications to build a system while others will explore an outsourcing option. Similar differences arise from the nature of the solution. A Web design project may require a different set of activities than a data warehouse project. The chapter covers the activities for an outsourcing solution—purchasing or leasing an application or service from a vendor. The activities for building a new system solution are discussed in Chapter 11.

The team following an outsourcing option may execute some or all of the following steps:

- Select an outsourcing model.
- Refine the requirements.
- Build a features comparison matrix.
- Specify an evaluation method.
- Write an RFP, if appropriate.
- Collect potential solutions.
- Evaluate potential solutions.
- Select, justify, and demonstrate a recommended solution.

With all outsourcing options, the team must decide on an appropriate framework or model for the outsource action: a commodity model or an association model. With commodity outsourcing, the team focuses on defining the desired features and developing methods to measure them. The purchaser tries to negotiate the best available price for the desired feature set and verify that the vendor provides the agreed-upon performance. For purchases in the association category, the team wants to examine such issues as vendor stability, vendor integrity, upgrade plans, policies and practices, ongoing product support, and warranties. Association contracts benefit from careful attention to dispute resolution procedures and personal relationships.

Once the team and the client come to an understanding on the appropriate outsourcing model, the team refines the specifications in sufficient detail to evaluate the suitability of various products and vendors. Clearly, the more expensive the contract, the more detail is warranted. The output of this work normally appears in a requirements narrative. If the product or system has a significant cost, most organizations require some sort of competitive bidding process using a request for proposal or RFP. The RFP process provides qualified vendors with an opportunity to bid under open and fair competition with understood criteria.

To outsource, the team communicates the desired features of a solution to the potential vendors. Mandatory features are ones that the client believes must exist in a solution. A set of mandatory features will exist for all purchased products. Desirable features are ones that the client wants, but the client may decide to accept a product without them. The features defined in the requirements may refer to

functional, operational, and vendor roles. Functional features describe what the solution must do, the client requirements. Operational features describe how the system meets the client's needs and can include options, maintainability, architecture, and other properties that affect how the system will operate. Vendor features describe such vendor characteristics as stability, integrity, support, and reputation. For association-type outsourcing, vendor features may weigh heavily in the evaluation.

Once the team and client identify the system features of interest, the team formulates a process for evaluating the information or bids received from vendors. Normally, the team prepares a features matrix or table to summarize and compare the results. Three commonly used ways to assign measures of merit are economic value, rating points, and features list. The client should determine the preferred approach. With economic value rating, the team assigns a dollar value of each solution/feature box. Many organizations prefer to use a rating point scheme (to assign points) to measure how well each product satisfies the feature.

The client may ask the team to provide only the evaluation framework or the results of the evaluation or a recommendation. When asked to recommend a specific product, the team's recommendation should consider all of the concerns raised by the client and not just look for the "best" score. The team can review the critical features of the recommended solution, organize the features in a clear way, and discuss the rationale for the recommendations as discussed in Chapter 9.

Key Terms

application service provider (ASP), 346	evaluation metric, 341	product flexibility, 349
association model, 341	features list, 363	rating point, 364
bid, 356	features matrix, 341	request for proposal (RFP), 341
commodity model, 341	functional features, 344	requirements, 343
comparison matrix, 341	interoperability, 346	security, 346
contract compliance, 350	maintainability, 346	software license, 347
data features, 344	mandatory features, 343	usability, 345
desirable features, 343	operational features, 344	vendor features, 352
economic value analysis, 363	outsourcing, 340	vendor stability, 352
	performance, 345	
	process features, 344	

Review Questions

Answer the following questions regarding these topics.

1. Outsourcing models.
 - a. What is an association outsourcing model? What is a commodity outsourcing model?
 - b. How do the contracts for these two models differ?

2. Requirements and features.
 - a. What is a mandatory feature of a system? Give an example from GB Video.
 - b. What is a desirable feature of a system? Give an example.
3. Types of features.
 - a. From GB Video give an example of a mandatory operational feature, a functional feature, and a vendor feature.
 - b. From GB Video give an example of a desirable operational feature, a functional feature, and a vendor feature.
4. Licenses.
 - a. Describe four types of copy restrictions common with software licenses.
 - b. What are the license provisions for Microsoft Windows?
 - c. Your client needs software to control a customer database with information that will be used by your mailing, billing, and sales departments. For the license, will you need a single station, single user, network, or site license? Justify your answer.
5. Features analysis.
 - a. What are three common ways of evaluating features?
 - b. How does each one work?
 - c. When would a feature list evaluation be appropriate?
 - d. Who determines weights in a features matrix? Who evaluates scores?
6. Solutions.
 - a. What are the common ways of discovering candidate solutions?
 - b. When might a client select a solution that does not have the “best” score?
7. Request for proposal.
 - a. What is a request for proposal (RFP)?
 - b. What are the five sections included in most RFPs?
8. RFP contents.
 - a. What is the relationship between the proposed system analysis (see Chapter 8) and an RFP?
 - b. Why should you not include the weighting scheme in the published RFP?
 - c. Is an RFP always necessary? Why or why not?
9. Outsourcing.
 - a. If your client has detailed specifications on which they are not willing to compromise, would you want to outsource? Explain your answer.
 - b. If a new product lacks one or more desirable features, should the team automatically dismiss the product? Explain.

Critical Thinking Exercises

Individual Exercises

1. Your client owns a gym and wants a system to keep track both of customers who have memberships and of potential customers.
 - a. Prepare an RFP for the system.
 - b. Provide an evaluation framework for the client.
 - c. Prepare a features analysis for the gym system.
2. How might you determine that you have identified the major potential vendors for a solution?

3. You have been asked to acquire a computerized system to automate the process of getting appointments with a college advisor for academic advice.
 - a. What are mandatory features for this system?
 - b. What are desirable features for this system?
 - c. How will you evaluate these features?

Group Exercises

1. Use GB Video as an example. Describe how you would evaluate each of the functional features listed in the text in such a way that you could defend the process to an auditor.
2. Review the Motor Vehicle Pool exercise from Chapter 3 (Creative Thinking Group Exercise 4). Write an RFP for the university to purchase a computer system to automate the rental process.

Chapter Eleven

System Design

Chapter outline

Introduction

A System Design Framework

Physical Infrastructure

Organizational Infrastructure

Infrastructure Example

Specifying Data Structure

Relational Schema

Metadata

Other Data Schema

Specifying Processes

Program Structure Charts

Physical Data Flow Diagrams

Process Model Metadata

Module Design

Module Specification

Pseudocode

Input Data

Transform Data

Operate on a Data Store

Perform a Procedure

Flow of Control

Output Data

Metadata for Module Logic

TIPOT Charts

Dialog-Driven Systems Design

Page Navigation Maps

Page Action Maps

Page Navigation and Action Map

Metadata

Data Warehouse Design

Dimensional Models

Data Warehouse Metadata

The Extraction-Transform-Load Process

Summary

Key Terms

Review Questions

Critical Thinking Exercises

Individual Exercises

Group Exercises

References

INTRODUCTION

Information system design focuses on program architecture and its interaction with the physical and organizational infrastructure in which the programs will operate. Unlike the conceptual solution requirements discussed in Chapter 8, which are technology independent, system design addresses logical and physical issues within the structure of specific technologies and organizations that the team plans to employ. As noted earlier, organizations increasingly purchase solutions. However, sometimes no purchased solution meets the requirements at an acceptable price or policy dictates in-house development. On these occasions, the team proceeds to the system design phase.

System design builds directly on extensive prior work and, as always, the team may wish to revise some of the prior work as more information is acquired. The team brings the results from the project definition, proposed system requirements and the alternatives, and evaluation and recommendation activities into systems design. These results include the following:

- An updated RD plan.
- Project definition materials that may include the statement of work, strategic alignment, desired features, constraints, and a current situation narrative with EDMs and DFDs.
- Proposed system requirements that may include a conceptual data model, modified data flow diagrams, and metadata for the CDM and MDFDs.
- A recommendation to build or create the system design for a specific alternative.

In the traditional SDLC model view, the team performs system design after the completion of the requirements determination phase. As noted in Chapter 9, real-world teams may perform system design concurrently with requirements specification. With prototyping, a team may carry out system design by building the prototype without any formal requirements determination and then using the prototype to define or refine the requirements.

The selection of the design approach for a given system is an art and often depends on personal preferences and/or organizational policies. Some people prefer procedural languages and design while others believe that object-oriented design works better for many applications. Language choice tends to drive part of the design approach. For example, such procedural languages as COBOL and FORTRAN are structured around the executions of subroutines and statement groups while such GUI languages as Visual Basic are organized around forms and controls.

The goal of system design is to create code and/or documentation with detailed specifications that will enable programmers to write and maintain correct and complete code for the proposed system. Many student field projects that select the build option create documentation and not code, sometimes because of time constraints, and at other times due to the client's preference for IT staff members to write the code. Good design documentation should communicate to an *experienced developer* the information that he or she needs to write and/or

maintain the code. The documentation specifies fully how the program works and what it accomplishes in an easy-to-follow format. The content of the documentation depends both on the system content—data, process, and physical and organizational infrastructure, and on the design approach selected by the team. Design documentation matches the requirements of the program to the specifics of database engines, operating systems, programming languages, access controls, and a host of similar items. Good design and documentation work together to simplify the coding and maintenance process.

System design utilizes the most detailed models of the systems solution process. As noted in previous chapters, a model provides a simplified and stylized version of the system that captures part of the essential features of the final operational system. Each specific model, for example, text descriptions, lists, charts, tables, DFDs, ERDs, relational schema, structure charts, pseudocode, code and other tools, focuses in-depth on part of the system and leaves out other parts. Because system design takes place at a detailed level in a specific technological and organizational environment, the appropriate models depend on the environment. For example, such process models as program structure charts may work well for applications implemented in procedural languages. Other models may better represent such applications as Web sites and data warehouses or such approaches as object-oriented design and object-oriented programming languages. In other words, the appropriate design documentation for a system consists of the set of models that best capture the critical features of the system. This chapter tries to illustrate several different system design approaches.

A SYSTEM DESIGN FRAMEWORK

The content model in Chapter 1 identifies the major content areas of an information system as data, process, and physical and organizational infrastructure. During system design, the team reviews each of the content areas and adds content or detail to arrive at a complete design that when operational will solve the problem posed by the client. These system design activities may include preparation of detailed design specifications for the following:

- **Physical Infrastructure.** Redefines the hardware and software as appropriate to facilitate performing the activities of the new system. For example, the new system may require some combination of new workstations, servers, telecommunications, data storage, operating system, database engine, or Web software. Or, the client may require use of the existing infrastructure.
- **Organizational infrastructure.** Modifies the organization as needed to match the functioning of the new system. Often, the new system requires the team and/or client to define the people who have the authority to create, update, retrieve, or delete data. The team also may need to prepare training plans for users, operators, and maintainers.
- **Data.** Converts the CDM into a data design. For example, establish a new database in the database engine, create a data schema of tables or files, and specify keys and indexes.

- **Process.** Specifies the detailed system logic. For example, prepare a program structure chart and write pseudocode for each program module and/or write the actual code for a prototype.

In summary, the outputs of the system design process may include

1. Physical and organizational infrastructure changes for the proposed system.
2. A data schema, often a relational schema.
3. Metadata to define the tables or files and attributes, columns or data items.
4. An appropriate process model, perhaps a program structure chart or module navigation map that describes the structure of the program.
5. Pseudo and/or actual code for each module in the program.
6. A prototype for the proposed system.

In addition to the central goal of meeting client requirements, good system design addresses such additional goals as:

- **Maintainability.** Good system design strives to produce a system to make it easier to make changes after the code has been developed. Good code structure and proper documentation make major contributions to supporting **maintainability**.
- **Inexpensive trial solutions.** Modern design techniques and tools may allow the development team to demonstrate trial solutions to clients and users with only small or moderate extra coding and revision expenses if clients and/or the team discover that parts of the trial solution are unacceptable.
- **Workload allocation.** A good design supports the allocation of code development to multiple individuals. Writing computer code is fundamentally an individual exercise. In order to allow more than one person to contribute at the same time, the team must partition the development requirements into units that interfere with each other as little as possible. Workload allocation is essential in larger projects to achieve reasonably short development times.

During the 1970s and 1980s, IT professionals, guided by the insights of Yourdon and Constantine, 1986, and DeMarco, 1978, developed techniques for program design that included such principles as **structured code, module coupling and cohesion**, and the translation of data flow models into program structure charts. These principles addressed the requirements of large transaction systems that ran on large mainframe computers, which were the predominant form of systems constructed in the period. The systems automated and updated manual processes or earlier generations of computer-based versions of core operating and administrative support functions in organizations. The systems typically used COBOL as a programming language and ran in batch rather than interactive mode.

Yourdon, Constantine, and DeMarco recognized that the primary maintenance problem for 1980s mainframe systems arose from the use of “go to” structures that produced very complex flows of logic and control and introduced

unexpected interactions between code changes in one part of a program with the execution of code in other parts of the program. Data flow and structure analysis developed (1) to plan and track control flow and (2) to avoid the unanticipated consequences that arose from the use of global variables in transaction systems driven by internal triggers and complex control flow. For these systems, maintainability was improved by structured control sequences that use only sequence, iteration, and selection to determine the execution sequence of cohesive and decoupled modules. Modern programming languages and approaches continue the principle of this structured approach in a variety of different implementations.

For example, relational databases (RDBs) and similar database mechanisms can reduce maintenance. Adding attributes to a table in an RDB to support a new program has (almost) no impact on existing programs. Changing a file structure in a COBOL program environment required program changes in all of the programs that used the file. Such languages as SQL, Visual Basic, and JAVA, have highly modular or object type structures that localize the impact of changes and reduce maintenance costs.

Programming guides contain a number of other suggestions for maintainability, many of which date back several decades. For example, good practice specifies that parameter values, for example, tax rates and discounts, always appear in tables and never in the actual code statements. When parameter values appear in the code, changes require reprogramming that can lead to a maintenance disaster. A different type of practice specifies that modules should have only one entry and one exit.

Physical Infrastructure

Physical infrastructure consists of such elements as database management systems, computers to perform the roles of workstations and servers, telecommunications, operating systems, middleware, programming languages, and development tools. The client's standards, procedure, and policy manuals may set forth guides and constraints for system design. For example, the client may specify a brand and configuration of PCs for users or may specify a database management system (DBMS) to serve as the database engine for all new applications. Or the client may restrict new development to one or several programming languages or tools.

As part of the narrative on the current situation, the team defined the current development environment and constraints established by the client. For some projects, the constraints may allow the team little or no choice with respect to the physical infrastructure for the proposed system. For example, the client may require the team to use existing hardware and software. In other projects, the team may receive agreement to make minor modifications, for example, to add Microsoft Access to the existing portfolio of software on an existing computer.

Occasionally, the team may select much of or the entire infrastructure for a project. Because client/server and net-centric architectures are highly modular, the client may wish to acquire new hardware and software for the proposed

application. If the client is not an IT person, the team should ask the client for permission to talk with an IT staff person to assure that the proposed infrastructure fits within company policy and standards.

The team prepares a comprehensive description of any proposed changes to the physical infrastructure for the client. The description may include a text summary of changes and the following information as relevant for each new item of infrastructure:

- Complete specifications for the item of infrastructure.
- Name, address, and contact information for the vendor(s).
- Price or cost, if available.
- Warranty or service agreement terms.
- Installation cost, method, and requirements.
- Maintenance cost and options—client, vendor, or third-party maintenance.

Software

Software specifications may address the operating system, database, and application languages. When feasible, a good process to follow is to select the application language or package program first, select a compatible database engine, and then select the operating system that works best. Specifications should include recommended tools for analysis or code generation. Some tools are language or product specific, for example, the Oracle toolset is designed to work with the Oracle database engine.

Hardware

Ideally, the team wants to select the software first and then select hardware that works well with the software. Server specifications normally include a make, model, and other relevant specifications including processor speed, number of processors, memory size, disk storage capacity, number of disks, caches, and network interfaces. Workstation specifications add the monitor specifications and such special input/output (I/O) devices as bar code readers. Routers are a common piece of hardware in many network systems.

Network

For those systems that operate over a **network**, the key issue addresses how the new solution will impact the network. Larger systems may require new links or components. The design team should specify the workstation, local area network (LAN), server, and network “cloud” requirements for the system.

Security

Security features may including firewalls, virus checkers, and backup requirements.

Organizational Infrastructure

During design, the team identifies and describes any desired organizational changes. For example, the new system may eliminate or reassign some of the

tasks performed by people in the current organization or may change the authority of people to create, update, delete, or retrieve information in the database. Changes in job duties or authority tend to cause anxiety. Clear identification of changes at this stage allows for appropriate training and counseling of the affected employees.

Before proceeding with system design, the design team may wish to identify clearly and/or review the following roles for the proposed system:

- **Clients.** The people or organizational unit responsible for providing the project team with the requirements for the proposed system. The sponsor is a client with the authority to request, control, accept, and revise the system. The sponsor may provide or control funding for the steps needed to complete the project. The sponsor may exercise or delegate authority and responsibility for implementing, training, operating, auditing, and maintaining for the system.
- **Users.** The people or organizational unit responsible for providing input to and/or using output from the system.
- **Data owners.** The people or organizational unit responsible for specifying data access controls, including, who can retrieve, create, update, and delete data and databases.
- **Operators.** The people or organizational unit responsible for running the programs in the system and for the correct operation of the system. Normally the operator follows rules agreed to by the client for running, restarting, and stopping the system.
- **Maintainers.** The people or organizational unit responsible for making changes approved by the client to programs and procedures for the system.

One person or unit may function in multiple roles. For a small system, one person might serve as client, data owner, and user, and another person as operator and maintainer.

Infrastructure Example

An example of the physical and organizational infrastructure for the proposed system for GB Video appears in Figure 11.1.

SPECIFYING DATA STRUCTURE

At this stage, the team designs the specific data schema for the proposed system. The team converts the conceptual data model into a data schema of relational tables, objects, other kinds of tables, flat files, or other forms. If the team plans to utilize a relational database management system (RDBMS), the team prepares a relational schema. Some CASE tools, for example, Oracle, with major help from the analyst, will convert a CDM (in Oracle notation using first normal form) into the corresponding relational table schema. With a flat file mechanism, the team specifies the file format using COBOL or other appropriate conventions.

FIGURE 11.1 Proposed GB Video Infrastructure

Physical Infrastructure

At present, the GB Video retail video rental and return environment operates manually. To implement the proposed system, the team recommends that GB purchase the following or equivalents:

Each of the three current stores will require two workstations (for a first version total of six) and one cable modem. Specifications for a recommended workstation and modem appear below.

Gentex Retail terminals Model 3301, \$2,499 each, full one-year warranty with in-office service Monday–Friday from 8:00 a.m. to 10:00 p.m. Includes MS Windows XP Professional, PC, 17" monitor, bar code scan gun, credit card reader and printer. The terminals are connected within the store via twisted-pair and RJ45 connectors to the cable modem. Vendor: Retail Systems, Inc., 7731 Main, Norman, OK 77019, Tel. 405-555-0768, Customer Rep: Tom Jones. After first year, a 14 × 5 in-store maintenance contract is available from Retail Systems at a current cost of \$10 per month per terminal.

Fox Cable will provide each store with a high-speed cable connection including a modem and installation for \$49.97 with a one-year contract. Service is \$37.95 per month per store including cable and modem maintenance and Internet access. Each modem will connect to up to three workstations (more with a separate router and additional monthly fees). Vendor: Fox Cable Business Services Division, 755 Lost Lane, Moore, OK 73170, Tel. 1-800-555-1234.

The team proposes to locate two servers at headquarters. These servers will connect to the existing router and Fox cable connection. Server specifications appear below. Retail Systems will purchase and install the servers (included in the price below) and make sure that they work correctly with the retail terminals. This server configuration should allow GB to expand to as many as 10 branches without upgrading.

Delta Power 2000 Servers. \$8948 total including software. Includes network interface, high-speed link, dual processors, 100 GB hard drive, 1 GB of memory, and Windows NT 5.2, SQL server 4.1, and Visual Basic 5.5 for application development. One year in office warranty 24 × 7; warranty service provided through Retail Systems. All transactions are shadowed on both servers so that full service is available as long as one is running. Vendor: Delta Systems, 1347 Boyd St., Houston, TX 72211, Tel. 1-888-555-4000. After the first year, maintenance is available from Retail Systems on a per call rate of \$30 plus \$50 per hour (one hour minimum) plus part costs. Extra charge of \$50 for weekend or evening service calls.

Organizational Infrastructure

The GB Finance Director (currently Sven Pasperson) in headquarters serves as the client point of contact for the new system. All requests for changes will go to the finance director. No direct changes in the store organization are required. All of the

people in a branch continue to report to the branch manager. Central Purchasing will continue to make decisions on the titles and number of video copies stocked at each branch.

Some changes occur with respect to location, ownership, and control of data. With the old system, each store owned and held physical possession of the data relating to rentals at the store. With the new system, the data on rentals and returns are stored on a server at GB headquarters. No data are stored at each branch. People at the branch can create, retrieve, and update customer, rental and return data but cannot delete it. Only the store operations manager at headquarters can delete data.

People at each branch can retrieve all data generated at any branch. A customer can examine and change his/her customer data at any branch and can inquire about the status of any open rental. No data are sent to Accounting. Instead, Accounting has access to the data tables for the rental system and the accounting system retrieves data as needed. The IS person at headquarters will operate and maintain the system. The organizational assignments have been discussed with and agreed to by all parties. No organizational problems or issues are anticipated.

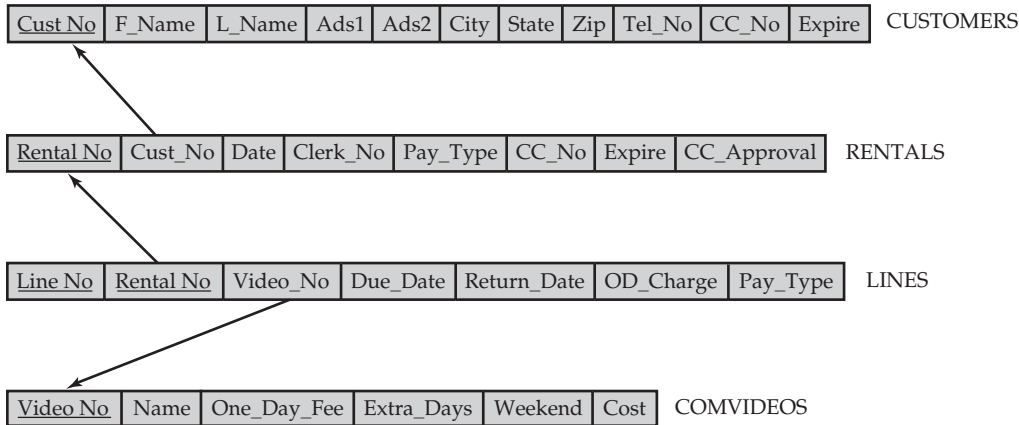
Page 2

Relational Schema

Relational databases are widely used. Enterprise resource planning (ERP) systems from such providers as SAP, People Soft, and Oracle recommend storing data in and retrieving it from relational database management systems (RDBMS). Many other package programs operate with an RDBMS. Because of portability and flexibility concerns, a majority of organizations design new applications for an RDBM system. However, an enormous number of nonrelational data stores remain in use for special applications and legacy systems.

If the team uses a relational database, the team may prepare a relational schema with the following rules:

- The proposed system CDM provides the structure for the relational schema. The schema should fully represent the complete data structure from the CDM. If the team decides that physical database considerations warrant a schema structure that differs from the CDM structure, the team should document the reasons in the system design report.
- Set notation, table-heading formats, or table diagrams can represent graphically the table, column, and relationship structure.
 - The analyst may use the entity name or the plural of the entity name on the CDM for the table name. Column names normally correspond exactly to attribute names.
 - The diagram explicitly should identify primary keys. A note on the diagram or in the text defines the convention used to identify keys, for example, the column name for a primary key is underlined.

FIGURE 11.2 GB Video Relational Schema (in column heading format)

Note 1. Primary keys are underlined.

Note 2. For data integrity control, Purchasing does not want the rental system to access their primary tables of Video and Title. Instead Purchasing will replicate the relevant data as needed in a view or table for the Rental System called COMVIDEOS. The denormalized table COMVIDEOS in the relational schema contains the combined attributes of the Video and Title entities in the CDM except that the attributes “Title No” and “Vendor No” are omitted in the table prepared by Purchasing. The combined table results in a simpler data structure and possibly, a faster response for the rental system.

- Referential integrity arrows link *from* each foreign key *to* each primary key. The arrows implicitly identify the foreign keys. Some analysts explicitly identify the foreign keys by placing (FK) after the column name or using some other convention.
- The team can use a CASE tool, drawing tool, MS Word, or similar tool to prepare clear, neat printed diagrams.
- Relational schema always must observe first normal form (1NF) and contain only a single value for each attribute in each row.
- If the schema is not in third normal form (3NF), the team should explain briefly the reasons to denormalize.

A sample relational schema in table heading format for GB Video that follows the aforementioned rules appears in Figure 11.2. In the schema, the entities Title and Video on the CDM in Chapter 8 appear combined in to a single table. The diagram contains notes explaining the convention for identifying primary keys and why the schema is not in 3NF.

Metadata

As a critical part of the data design, the team selects metadata for the tables and columns. *Data items* consist of the attributes or column headings in relational tables or the data element names in flat file records. The team may need to define data items for use in input, output, and transformation operations that are not part of, or placed in, any data store. For example, the program may compute the total cost for a rental. Total cost is a data item used to prepare and print an invoice but is not part of any data store and thus is not an attribute or data-element. Data items that are not part of a store are described in a separate section called “Data Items” following the attribute or data element descriptions.

In practice, the metadata for a table or file and data items can include dozens of categories. The categories selected for the system design deliverables may depend both on (1) those categories included in the database engine and (2) the standards and policies of the organization. A minimal set of table or file metadata might include:

1. A short text description of the table—normally identical to and copied from the CDM entity description.
2. Retention policy—how long or under what circumstances is each record or instance or row in the table retained. Good retention decisions balance control of table size against the availability of data for operations and historical analysis. For such entities as Item, the retention policy may be obvious. The record for an item is retained as long as the item is stocked or sold, and deleted or transferred to a history file when the item no longer is stocked. Retention policy is most important when files tend to increase as a result of activity. Transaction files for sales, rentals, and purchases exhibit this behavior.
3. File size—how many records or rows will the file or table contain and how many megabytes of storage are required? As file size grows beyond the forecast number, physical storage requirements often pose less of a problem than processing time. Additional physical storage is easy to add and relatively inexpensive. But increased file sizes may cause a slowdown in file processing activities that is more difficult to remedy. As noted above, file size sometimes can and should be controlled by retention policy.
4. Authorization—identifies who is allowed to create, update, retrieve, and delete records or rows in the table. Unless authorization is defined clearly, organizational conflict and loss of data integrity may result.

A minimal set of data item metadata includes:

1. A short text description for each data item—normally copied from the attribute descriptions in the CDM if relevant.
2. Optionality—whether the user is allowed to omit a value for the data item when first creating a new row or record. Optionality is important for data integrity. By using optionality the team can assure that the primary key and other mandatory or required attributes are present at the time a record first is created.
3. Data type—a specification that allows the physical system to store data efficiently. The team normally will use the **data types** provided by the database engine specified in the infrastructure for the proposed system or in the proof of concept model, for example, Access, MS SQL Server, and others. In Access, the available data types include text, number, date/time, memo, and yes/no. The team should reference the source of the data types used in their system design documents.
4. Size—the maximum length to allow for each data type. Some data types, for example date and integer, possess a size determined by the database engine. Part of the size description for a decimal number includes the number of decimal places.

A sample set of metadata for the GB Video relational data schema in Figure 11.2 appears in Figure 11.3. Note that in this example and in most cases, the table

FIGURE 11.3 GB Data Schema Metadata**Table and Column Descriptions**

Note: (PK) shows a primary key and (FK) a foreign key. Data types are from MS Access.

CUSTOMERS. Contains all the available information about each customer who has made a transaction in the last year.

Retention. The records for customers who do not make a transaction for one year are deleted.

Size. The database will initially contain 3,000 customer records and require about 1 MB.

Authorization. Clerk: create, update, and retrieve. System Manager: all.

Data-Item	Description	Optional	Type	Size	Decimal
Cust_No	A unique identifier assigned to each customer (PK)	No	Text	10	
F_Name	First name and middle initial if any	Yes	Text	15	
L_Name	Last name	No	Text	30	
Ads1	Street or box address	No	Text	30	
Ads2	Apartment number or other as needed	Yes	Text	30	
City	Name of city	No	Text	20	
State	State id code	No	Text	2	
Zip	Zip code	No	Text	9	
Tel_No	Telephone number	Yes	Text	10	
CC_No	Original credit card number	No	Text	16	
Expire	Expiration date on the original credit card	No	Date		

RENTALS. Contains the header information on each rental transaction.

Retention: Transaction instances are removed to backup storage 15 days after a transaction is complete.

Size: Approximately 1,500 records and less than one megabyte.

Authorization: Clerk: create, update, and retrieve.

Data-Item	Description	Optional	Type	Size	Decimal
Rental_No	Unique identifier assigned to each rental (PK)	No	Auto-number		
Cust_No	The customer for the rental (FK)	No	Text	10	
Date	Date of the rental	No	Date		
Clerk_No	Employee number of the clerk entering the rental	No	Text	3	
Pay_Type	Cash, check, or credit card	No	Text	1	
CC_No	Credit card used to rent	Yes	Text	16	
Expire	Expiration date of the credit card used for the rental	Yes	Date		
CC_Approval	Credit card approval code	Yes	Text	6	

LINES. Contains the information on each video associated with a rental transaction.

Retention: same as Rentals.

Size: Approximately 2,000 records and less than one megabyte.

Authorization: same as Rentals.

Data-Item	Description	Optional	Type	Size	Decimal
Line_No	Unique identifier assigned to each line (PK)	No	Auto-number		
Rental_No	The rental number that this line belongs to (FK)	No	Long Integer	10	
Video_No	Video rented on this line (FK)	No	Text	14	
Due_Date	Date video is to be returned	No	Date		
Return_Date	Actual return date	Yes	Date		
OD_Charge	Charge for days kept after due date if applies	Yes	Currency		2
Pay_Type	Method of payment for the overdue charge	Yes	Text	1	

COMVIDEOS. Contains information on each video. This database table is maintained and supplied to the rental system by purchasing.

Retention. NA.

Size. Approximately 500 records and less than one megabyte.

Authorization. Clerk: retrieve only.

Data-Item	Description	Optional	Type	Size	Decimal
Video_No	A unique identifier assigned to each video (PK)	No	Text	14	
Name	The title of the video	No	Text	30	
One_Day_Fee	First day rental fee	No	Currency		2
Extra_Days	Extra days rental fee	Yes	Currency		2
Weekend	Rental fee for Sat. and Sun.	Yes	Currency		2
Cost	Price GB paid for the video	No	Currency		2

Table Relationship Metadata

Relationship Description	Table	Foreign Key	References Table: Attribute
A customer may make one or more rentals	RENTALS	Cust_No	Customers: Cust_No
A rental may contain one or more lines	LINES	Rental_No	Rentals: Rental_No
A video copy may be rented on one or more lines	LINES	Video_No	Comvideos: Video_No

and column descriptions follow the CDM entity and attribute descriptions. The table relationship material describes in text form the table relationships on the data schema diagram.

Other Data Schema

Some projects will involve a data schema other than relational tables. For example, in a system that uses flat files, tables similar to those that make up the COBOL data division may define the data schema. The model selected should describe completely the data schema—the table, file, or object structure, and include a full set of metadata.

SPECIFYING PROCESSES

In this step, the team specifies both (1) the module or program structure and (2) the logic or code within each module or process. The way that logic is specified will vary from project to project. Design documentation normally uses physical models—models that directly map the system as it will be built. Module, table, and data element names, and calling and triggering sequences in the design documentation models should correspond exactly to the statements in the program code. The basic approach for process logic specification is:

- Represent the structure of the logic in an appropriate graphical form—program structure chart, physical data flow diagram, object schema, or page navigation map.
- Specify the detail logic within each module or procedure shown on the graphical representation in pseudocode or program code.

As noted earlier, at the logical and physical levels, the content and form of the system influences the selection of an appropriate process model. While process-driven systems remain common, many new systems are either **data-driven** or **dialog-driven**. Data warehouse and flexible reporting systems typically are data-driven. The process logic for these systems consists of “feed and read,” with simple modules that collect data from existing sources and straightforward reporting modules to present reports. The critical issues for these systems focus on organizing the data in a format that allows simple reporting logic. In a dialog-driven system, interactions with the user control the flow of the program. Such a process model as a program structure chart may not represent well the critical design issues for dialog-driven systems.

The next section of this material describes the use of program structure charts and physical DFDs to describe processes. Since most process models use the concept of modules, additional sections describe module design, use pseudocode to define logic in modules, and add structure to module metadata. Subsequent sections describe possible alternative process models for dialog-driven systems.

Program Structure Charts

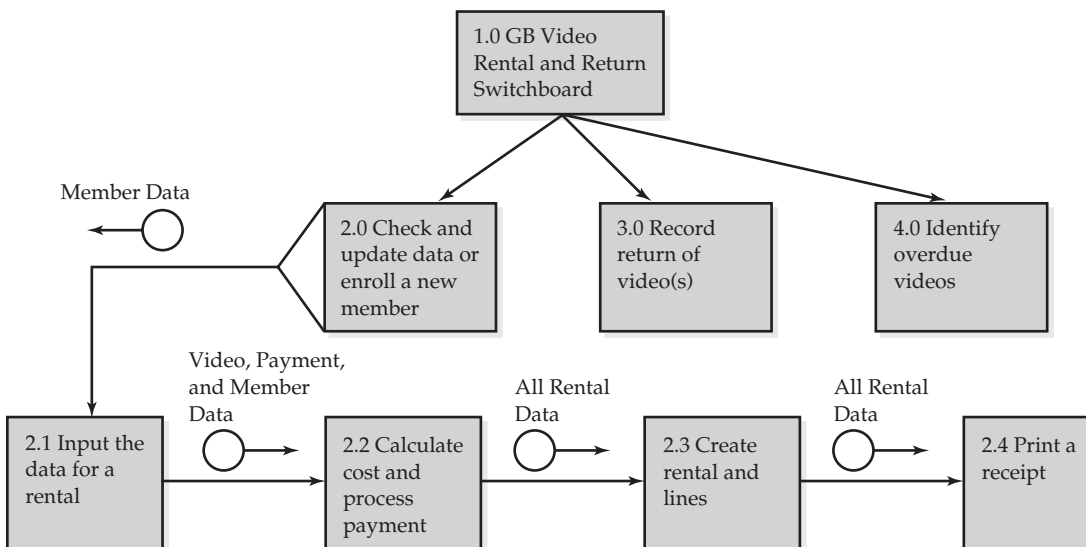
A **program structure chart (PSC)** can provide a graphical model of process structure for many programs, especially process-driven systems implemented in a procedural language. Each **module** in the proposed system is represented by a rectangle in the PSC. Each module receives a unique label and a description. In addition to the modules, structure charts show the triggering sequence for modules and the control flows and data flows between modules. To show trigger sequence, an arrow connects each module to the other modules that it may trigger. Sequence arrows do not receive a name. The control flow is specified by the detail logic inside each module. Detail logic for each module (discussed in a later section) defines the operation of the module. In a hierarchical design, each module has a unique path into it. Hierarchical designs tend to simplify debugging and maintenance more than architectures that allow multiple paths into modules.

Many variations of and symbol sets for structure charts exist. Some structure charts show control flags and data flows passing from one module to another. When control flags or data are passed between modules, a named arrow represents the flow. Arrows with an open circle on the tail represent data flows; arrows with solid circles indicate control flags. Each flag or flow receives a descriptive name that matches (as much as possible) the name of the element actually passed between modules. Typically these diagrams do not include data flows to or from data stores or externals. Physical data flow diagrams, covered in the next section, provide a way to show those data flows.

The boxes on a structure chart correspond one-to-one with code components in the final program. With such procedural languages as COBOL or C, the modules become paragraphs, subroutines, or functions. In such GUI-oriented languages as Visual Basic, modules may correspond to forms, controls, macros, or queries. The name of the box on the structure chart should match the name of the corresponding structure in the program code. In the metadata sections, pseudocode or structured English details the process within the module.

A sample program structure chart for the GB Video Rental and Return program appears in Figure 11.4. Note that modules shown on the chart resemble the modules in the modified data flow diagrams for GB Video shown in Chapter 8. Module 1.0 is a switchboard module that allows the system user to trigger or call modules 2.0, 3.0, and 4.0. For example, when a video is returned, module 3.0 is triggered. The arrows from one module to the next represent triggers. As requested by the client, the first rental module, 2.1, is triggered by the member module 2.0. The diamond attached to module 2.0 indicates a conditional action; the rental module is triggered only when a customer wants to

FIGURE 11.4 Program Structure Chart for GB Video



make a rental. Sometimes the customer just wants to become a video store member.

Since the arrows represent triggers, a special symbol of an arrow leading away from an empty circle, is used to indicate data flowing from one process to another. The arrow direction shows the direction of the flow. The name for the data flow arrow identifies the data in the flow. Many of the modules shown in Figure 11.4 contain multiple functions. An actual program structure chart probably would contain a number of additional modules. A later section describes module design.

Physical Data Flow Diagrams

Physical data flow diagrams (DFDs) show graphically the same module structure that appears in the program structure chart plus data flows to and from externals and data stores. Many programs consist of such modules as subroutines and functions that receive data from another module or external, modify the data, retrieve from and store data in data stores, and send data to another module or external. In these systems, the flow of data drives the execution of the system. Data flow diagrams capture the essentials of these systems.

Physical DFDs add more detail to the MDFDs discussed in Chapter 8. A physical data flow diagram describes how the final system should work. In a physical DFD, the modules on the diagram correspond to actual constructs in the code. Processes have the same name as components in the program and are sequenced in the order that they will execute in the program. Data elements in flows, data stores, and queries have the same names as they do in the actual database. Physical data flow diagrams also show the flags, audit trails, and error checks that are included the actual system.

Process Model Metadata

All of the objects on the process models are defined further by metadata. The exact metadata included depend upon the model used. Possible metadata follows:

- Data flows. The data items that make up each flow. The metadata for each data item appears in the data model.
- Data store. Defined by a table schema or a data division with their metadata.
- External. Brief description.
- Link lines. No metadata.
- Modules. The pseudo or actual code for each module.
- Menu. The table showing menu number, text, and hypertext addresses for the menu. *Note:* A menu is defined once although it may be used for multiple pages.
- Page. A brief description of the html text on the page or the actual text enclosed in quotes. A table can show each hypertext link on the page that is not part of a menu in the first column and the Web address for the link in the second column.
- Procedures. A brief description of the procedure plus the detail logic for each module in the procedure. Use an MDFD for complex procedures.

MODULE DESIGN

Several different names are used to refer to modules; the most common names are program module, code module, process module, and functional module. Normally code or program module names are used at the detail design level. Organizational functions names are used for modules at the requirements level. A program or code module contains a set of code, pseudocode, or logic statements that work together toward a common purpose in a program. A module may correspond to a box in a structure chart, or a module or several modules may contain the code for a radio button in Visual Basic or for a procedure in an object.

Identifying a good set of program modules for a program requires skill and careful thought. In general, a good module contains a high level of cohesion and a low level of coupling. The strength of the interrelationships between the logic statements in a module is called **cohesion**. In a functionally cohesive module, all of the statements relate to the performance of the same function. For example, one of the GB Video modules might focus on the function of inputting and recording the ID numbers of the videos that a customer wishes to rent. Note that GB Video is an event-driven system. In event-driven systems, actions by the system user are events that cause modules to execute. The logic statements for the functions associated with events also may possess temporal cohesion; they occur in time sequence without interruption by statements from any other module. Functional cohesion generally is the most important cohesion issue in designing effective program modules. Functionally cohesive modules simplify program design, testing, and maintenance.

The module structure in the program structure chart for GB Video is functionally and temporally cohesive. Each module at each level deals with a set of activities that relate to the same function and occur continuously in time, or one after another without pause or interruption. One easily can devise modules that do not show a high level of cohesion. For example, consider a module with the description, "Read in the video identifier from the scanner, create a rental record, and calculate the overdue charge for a late return of a videotape." Intuitively, most analysts would recognize that such a module seems like poor design. The set of logic for calculating overdue charges for the late return of a video clearly shows little cohesion with inputting video numbers and creating rental records for a rental—the two sets of logic statements relate to different functions and occur at different times.

Other forms of cohesion exist. For example, all the statements that utilize the same set of data may be placed in the same module to enhance communicational cohesion. With sequential cohesion; a statement requires the results produced by a previous statement to perform its function. To compute the tax on a rental, the program first needs to compute the total charge for the rental.

Coupling refers to the nature of the interactions between modules. Coupling can occur in a variety of ways. With data coupling, a relative weak kind of coupling, a module connects to another module only by the transfer of data or messages to and from the other module. In object-oriented design and subroutine calls, message coupling is the only form of coupling allowed. Data coupling can

cause a problem when data-sensitive actions or limits exist in a module. For example, if a rental line module is set up to handle a maximum of 10 videos per rental, the module may fail when the rental module sends more than 10 video IDs to it. If the module receiving the data will perform the expected actions and only those actions for all allowable values and sets of data, then data coupling presents few problems. In most modern applications, modules are coupled mainly by data.

Older systems may contain more complex and difficult-to-predict forms of coupling. When modules use or modify code from another module, coupling often causes problems. With hybrid coupling, one module changes the code in another module at run time. Systems with hybrid coupling are difficult or nearly impossible to maintain. An analyst trying to maintain the code may not know which statements actually will run when the code executes. A good design objective is to be able to maintain and modify modules without having to understand the logic of other related modules.

Module Specification

Structured module and object-oriented design try to increase cohesion and reduce coupling. However, no one knows how to design the “best set” of modules. As in many other parts of IT design, the analyst follows the procedure called *satisficing*; avoid clearly bad structures of modules and strive to find a structure that at least is satisfactory. Guidelines for structured module design at the detail logic level include:

1. A module performs one and only one major function, such as to charge credit card, prepare invoice, retrieve data, and update data. The statements within the module should show a high degree of functional and temporal cohesion unless another form of cohesion is more important.
2. Each module should work correctly for any data set that its trigger module(s) may send to it. The less data transfer the less chance for data coupling problems, but good design involves trade-offs. Retrieving data from a data store instead of receiving them from another module may reduce the incidence of data coupling problems but may increase processing time.
3. Every module contains logic defined by pseudocode or actual program code. To carry out their function, modules may (a) operate on data stores to retrieve, create, update, or delete records; (b) read input from such external sources, as keyboards or scanners; (c) send output to such external sinks, as printers or other systems; (d) perform data transformations; (e) call or perform subroutines or procedures; and (d) use sequence control logic to direct the control flow within the module or to select the next module to trigger.
4. A module should have one and only one entrance—the statement executed when the process is triggered. It may exit by transferring control to one or more other modules, or it may cause the program either to close or to wait for an external event trigger. A module may have sequence control logic at the exit point, that is, it may trigger one of several modules or none depending on internal conditions. Control transfers in one direction only: from the exit point of the trigger module to the entrance point of the triggered module.

5. The only communication allowed between modules consists of the data items passed from the trigger module to the triggered module at the time of the trigger event. The data may include flag or switch values. Flags and switches increase the potential for data coupling problems and should be used with caution.

Statements from a programming language or pseudocode specify the logic within a module. When statements from a programming language are used, the analyst specifies the specific language and version. The actual code statements should follow the format and conventions of the language. The analyst can substitute pseudocode or a brief description for actual code in complex or stub modules to be coded at a later time.

Analysts use several different conventions to govern which modules may contain code. With one convention, code may appear only in the lowest level modules, those modules with no children or descendants. These modules are called *basic functional modules* or *elementary modules*. With this convention, additional elementary modules are added as appropriate to contain switchboards, menus, and setup instructions. Other analysts find it more convenient to allow modules that are not elementary modules to contain switchboards, menus, and setup instructions.

Pseudocode

Pseudocode or its close relative, structured English, offer a generalized programming language to specify the detail logic associated with each module. Pseudocode consists of a set of instructions or statements similar to those found in programming languages. As in a programming language, the pseudocode statements in each module must explicitly, completely, and unambiguously define the actions performed by the module to transform inputs into the required outputs including links to subroutines and other modules.

The actions or statements available in pseudocode include:

1. Read input data from an external source, such as a keyboard or scanner.
2. Transform data—assign a given or computed value to a data element.
3. Operate on a data store—retrieve or update attributes in an existing record and create or delete records.
4. Call or perform subroutines or procedures.
5. Use a control to execute statements, macros, modules, or subroutines.
6. Manage the flow of control. Unless otherwise specified, control within a module is assumed to flow sequentially, from each statement to the following statement. Use conditional logic to control any other sequence of actions—if, case, or do statements. Control flow in event-driven systems is managed by the user activating a control event.
7. Write output data to an external sink. Output can include print a card, document, or report, send data by EDI, and so on.

While some general conventions apply, much of pseudocode is flexible. Companies and/or analysts may develop their own versions as needed. The material to follow sketches out some pseudocode statements for use in a project. Team members should be familiar with the actions performed by these statements from

programming courses. Pseudocode may use standard programming conventions, for example, separate data item names by commas. However, punctuation is not critical in pseudocode as long as the meaning is clear. For example, either periods or semicolons can mark the end of a statement. For a number of situations, an analyst may wish to invent or expand on core pseudocode to represent some complex action in a simplified form.

The following sections illustrate pseudocode format and general conventions for use in the field project. This material assumes that team members are familiar with programming languages and conventions. Review an introductory programming language book if any of the statements below are unclear.

Input Data

The general form of the pseudocode statement is, Read (data item names) [From (source name)]. Any part of a statement in square brackets [xxx] is optional. The following pseudocode statement reads in the customer number for a video rental: Read I_Cust_No [From Scanner].

The “I” before the attribute name is an optional convention to identify the fact that this data item, which contains input data from an external source, is not an attribute or data element retrieved from a data store. The programmer may use the Read statement only to bring in data from an external source; you may not Read from a data store.

Transform Data

The Set statement sets a data item equal to the value of an expression. The format of the pseudocode is, [Set] (data-item) = (a given or computed value of an expression). The word *Set* is optional but helpful to clarify that this statement sets or assigns a value to the data item on the left-hand side of the equation. For example, the statement “Set Tax-Rate = 0.07” will set the sales tax rate to a given value, 7 percent. The statement “Set Tax = Tax-Rate * Total-Cost” sets the value of tax equal to a computed value: the tax rate multiplied by the total cost. All standard functions—average, maximum, total, and others—may be used in pseudocode.

The analyst may use or invent a nonstandard function described in English, as long as the analyst provides adequate detail on what the custom statement does. For example, the statement “Set Cust-No = (next unused customer number in the sequence)” represents the automatic generation of customer numbers more clearly than “Set Cust-No. = (next).”

Operate on a Data Store

SQL provides the basis for the pseudocode to specify data store operations, to create, delete, retrieve, or update a row or record. The analyst may wish to review SQL before trying to write pseudocode for data store operations. A pseudocode statement to retrieve data about a GB Video customer might be “Select * From Customers Where Cust-No = I-Cust-No.” This statement retrieves all the data for the customer whose member number was scanned and stored in I-Cust-No. Any valid form of the Select statement or any SQL statement may be used in the pseudocode.

Pseudocode allows the analyst to invent English phrases to simplify the code as long as the meaning is explicit and complete. For example, the system may provide for retrieval of a customer record by customer number, telephone number, or name—normally three different SQL statements. The customer number option retrieves the data for one customer; the others may retrieve data for several customers. Additional program logic is required to let the clerk select the desired customer from multiple choices. A shorthand expression for this process that is acceptable at the pseudocode level is “Select * From Customer Where” (the current customer may be identified by customer number, telephone number, or name).

All SQL statements may appear in pseudocode. For example, an Insert statement will create new rows or records. The statement, “Insert Into Rentals Values (I-Rental-No, I-Cust-No, I-Date, I-Clerk-No, I-Pay-Type, I-CC-No, I-Expire)” will take the values inputted to or generated in the GB Video rental process and place them in a new row in the Rentals table. The Delete statement will delete an entire row or record. An Update statement changes one or more values in an existing row or record. The statement, “Update Lines Set Return-Date = (current date) Where Video-No = I-Video-No and Return-Date = null” will update the record to reflect the return of a video.

Statements to create new tables or files may be, but normally are not, included in the pseudocode. The data schema and its metadata should provide an adequate system design specification for the actual code to create the tables. Some CASE tools will create the data schema from an ERD and (with some help from the analyst) create the tables from the schema.

Perform a Procedure

The Perform statement is used to execute a procedure—a named block of code inside the module, or to call a subroutine outside the module. Procedures inside the module allow the programmer to use the same set of instructions at more than one point inside the module or to simplify complex conditional expressions by separating out the statements in the procedure. A procedure is created inside the module by giving a set of statements a heading “Procedure (name)” where *name* is any unique procedure name, a name not used by another procedure. In the metadata for a module, the procedures follow the sequentially executed code block. The format is:

```

Procedure Name-1
Begin
Statement 1
. . .
Statement n
End

```

Subroutines consist of procedures used by more than one module. The descriptions for these procedures follow the module descriptions in a section called “Common Subroutines.”

The format of the statement to execute both internal and subroutine procedures is “Perform (name).”

Flow of Control

As noted earlier, control flows sequentially from statement to statement in the pseudocode unless changed by a conditional statement. Conditional statements in pseudocode include If, Then, Else, Do, and Class. The format for the If statement is “If (condition) Then (action) [Else (action)].” Action can consist of any pseudocode statement including a Perform. When a module wishes to pass control to another module, the statement is “Trigger (module ID number [{list of data-items}]),” for example, the statement “Trigger P2.2 {Cust-No, Rental-No}” may appear at the end of module P2.1. The flow of control and the values of the data items pass from P2.1 to P2.2.

Do statements appear in many forms. Three general forms for pseudocode are:

```
While (condition) Do
    (Actions)
EndWhile
```

```
Do
    (actions)
Until (condition)
```

```
Do For (index = initial To limit)
    (Actions)
EndDo
```

In Do statements, actions may consist of a set or block of pseudocode statements including Performs.

The Case structure provides an efficient way to show the choice among several alternatives. The format is:

```
Select (data-item)
    Case (value-1, action-1)
    . . . . .
    Case (value-n, action-n)
    Default Case (action)
EndSelect
```

The statement involves a set of linked values and actions. The action paired with the case value that matches the value of the data element is executed. If pushing the Member button, sets the data item choice = 1, Return sets choice = 2, and Overdue sets choice = 3, then the following pseudocode will select the appropriate GB Video process:

```
Module 1.0
Read Choice
Select Choice
```

```

    Case 1 Trigger P2.0
    Case 2 Trigger P3.0
    Case 3 Trigger P4.0
    Default Case Perform Error1
EndSelect

```

No data are passed in this example, only the flow of control. Note that the programmer may use embedded If statements to accomplish the above selection, but the Case statement offers an easier-to-follow form.

Output Data

The Write statement is used only for output to an external sink. The programmer may not write to a data store. The format is “Write (data-item list) {To (device)}.”

For example, Write Name To (member card imprinter) places the customer’s name on a member card that already contains both a printed and bar-coded member number.

Metadata for Module Logic

The metadata for each module in the program structure chart or the physical DFD describes the logic for the module. The objective is to provide enough detail so that a competent programmer can generate the actual code. The description may contain all or some of the following items:

1. Trigger for. An identification of all the events that can trigger or start the operation for this module. Such events as an action (e.g., a button click) on a data entry screen, a call, or message from another module, or a specified time or count condition may serve as triggers for a module.
2. Input data. All possible data inputs to the module and the source of the data: external, another module, or a data store. If an input form, diagram, or screen shot is associated with the input, the associated diagram may be referenced. Retrieval from a database may include appropriate code or pseudocode for the retrieval.
3. Process. Descriptions in structured English, pseudocode, or actual code of the processes that take place in the module.
4. Output. All output data and their destination: another module, data store, or external.
5. Triggered by. Identification of all the potential modules triggered by this module. The logic for determining which module is triggered appears in the process description.

Figure 11.5 shows a sample set of module logic with pseudocode for modules 1.0, 2.0, 2.1, 2.2, 2.3, and 2.4 of the GB Video program structure chart. The module logic carries out the processes shown in the program structure chart in Figure 11.4 and in the conceptual DFDs in Chapter 8 and uses the data structure of the relational schema in Figure 11.2. Many of the statements resemble actual code in a number of procedural programming languages. Some statements use brief

FIGURE 11.5 Metadata for Modules of the GB Video Rental System1.0 GB Video Rental and Return Switchboard

The system manager or operator triggers this module. The module contains the code to start and restart the system and to display the main switchboard that allows the user to select or trigger 2.0, 3.0, or 4.0

Read (function) from Keyboard

Select (function)

Case (1, Trigger 2.0)

Case (2, Trigger 3.0)

Case (3, Trigger 4.0)

End Select

End

2.0 Check and update data or enroll a new member

Triggered by 1.0

Input (I_Customer_status)

Select (I_Customer_status)

Case(Old, Perform Old)

Case(New, Perform New)

End Select

Procedure Old

Begin

Read I_Cust_No, I_Tel_No

Select * From Customer Where Customer:Cust_No = I_Cust_No or Customer: Tel_No = I_Tel_No

Display customer data

Input any corrections to customer data

Update Customer

End

Procedure New

Begin

Read (all attributes of customer except I_Cust_No)

Set I_Cust_No = (next unused customer number)

Insert Into Customer Values ()

Print "Member Card"

End

Display message "Select Yes to proceed with a rental; No to End"

Read (I_Rent)

If I_Rent = Yes, then trigger 2.1 and send customer data

End

2.1 Input the data for a rental

Triggered and sent customer data by 2.0

Generate (I_Rental_No, I_Date)

```

Set i=1
Perform
  Read (I_Video_No[i])
  Set i=i+1
Until no input
Read (I-Pay_Type)
If (I_Pay_Type) = 1, then Read (I_CC_No, I_Expire)
Trigger 2.2 and send video, payment and customer data
End

```

2.2 Calculate cost and process payment

```

Triggered and sent video, payment and customer data by 2.1
Perform Calculate_charge ( {Video_No}, i, Charge)
If I_Pay_type = 1, then Perform CC approval (CC_No, Charge, CC_Apv) else trigger 2.3
[Note: Calculate-charge and CC-approval processes to be coded later]
If approval denied, then End else trigger 2.3 & send all rental data
End

```

2.3 Create rental and lines

```

Triggered and sent all rental data by 2.2
Insert Into Rental Values ( )
Perform for j=1 to i Insert Into Line Values ( )
Trigger 2.4 and send all rental data

```

2.4 Print a receipt

```

Triggered and sent all rental data by 2.3
Retrieve the video titles from Title
Perform Print-receipt
End

```

English expressions that describe the desired action in place of statements that look like code. As noted earlier, pseudocode conventions will vary from organization to organization. Figure 11.5 uses the symbol {I_Video_No} to indicate the multivalued attribute or set of video numbers included in each rental. The code uses singular table names in place of the plural names on the relational schema.

Several of the pseudocode procedures are stubbed, for example, the procedure to charge a credit card. Normally, the credit card processing company supplies the interface specifications and/or code to use for this process. In the pseudocode shown, each module sends all the data it uses or generates to the next module. In an actual program, the programmer might declare the input and retrieved variables as available to all the modules of Process 2 and eliminate the need to keep sending data to each newly-triggered module.

Chapter 12 illustrates the steps necessary to implement either process- or dialog-driven designs for the GB Video system. The Microsoft Access code that implements the modules on the GB program structure chart in Figure 11.4 appears on this book's Web site (<http://www.mhhe.com/vanhorn>). While the actual code accomplishes the same functions as the pseudocode, the actual code illustrates a look and organization in accord with the specific technology conventions and structures of Microsoft Access.

TIPOT Charts

A **TIPOT chart** (Triggered-by, Input, Process, Output, Trigger-for) offers a structured way to document module logic. The TIPOT chart (pronounced as “teapot”) adds trigger logic, pseudocode, and more physical details to the traditional IPO charts discussed in Chapter 5 and is often used as a requirements documentation tool. The chart contains much of the same information as the metadata described in the previous section but uses a table format. Each row of the TIPOT table contains the data for one module. The columns of the table are described below.

1. Triggered by. The first column lists all of the possible events that the module in this row can be “triggered by.”
2. Input. The second column describes all required data inputs and the source of the data—external, another module, or a data store.
3. Process. The third column contains the process descriptions in structured English, pseudocode, or actual code.
4. Output. The fourth column describes all outputs and their destination—another module, data store, or external.
5. Trigger for. The fifth column identifies all the potential modules that the module in this row can serve as a “trigger for.”

Table 11.1 shows a TIPOT chart for the switchboard, member, and rental processes in the GB Video structure chart in Figure 11.4. The TIPOT chart corresponds to the metadata in Figure 11.5.

DIALOG-DRIVEN SYSTEMS DESIGN

In dialog-driven systems, the actions of a user determine, in large part, the sequence of module execution. Such standard process models as program structure charts and physical DFDs can represent adequately the processes in some event-driven systems, for example, the GB Video Rental and Return system. However, many field projects, for example, Web site and interactive database applications, involve relatively simple, straightforward processing options driven by user menu choices. The visual languages that support them (Visual Basic, DreamWeaver, etc.) are organized around screens with buttons and input data boxes on them to execute the actions the user selects. The basic modules for these systems consist of screens or forms and the actions associated with each screen.

Possible alternative process models for the design of these classes of systems include **page navigation maps** and **page action maps**. Navigation maps identify the screens and the way that the program can proceed from screen to screen.

TABLE 11.1 TIPOT Chart for GB Video

Triggered By	Input	Process	Output	Trigger For
External: Manager or operator	Function Choice Source: Keyboard	<p>1.0 <u>GB Video Rental and Return</u></p> <p>The system manager or operator triggers this module. The module contains the code to start and restart the system and to display the main switchboard that allows the user to select or trigger 2.0, 3.0, or 4.0</p> <p>Read (function) from Keyboard Select (function) Case (1, Trigger 2.0) Case (2, Trigger 3.0) Case (3, Trigger 4.0) End Select End</p>		P2.0 P3.0 P4.0
P1.0	Customer Request Data Source: Keyboard	<p>2.0 <u>Check and update data or enroll a new member</u></p> <p>Triggered by 1.0 Input (I_Customer_status) Select (I_Customer_status) Case(Old, Perform Old) Case(New, Perform New) End Select Procedure Old Begin Read I_Cust_No, I_Tel_No Select * From Customer Where Customer: Cust_No = I_Cust_No or Customer: Tel_No = I_Tel_No Display customer data Input any corrections to customer data Update Customer End Procedure New Begin Read (all attributes of customer except I_Cust_No) Set I_Cust_No = (next unused customer number) Insert Into Customer Values () Print "Member Card" End Display message "Select Yes to proceed with a rental; No to End" Read (I_Rent) If I_Rent = Yes, then trigger 2.1 and send customer data End</p>	<p>Member Data Destination: P2.1 and Screen</p> <p>CustomerData Destination: Customer Table</p> <p>Member Card Destination: Printer</p>	P2.1

TABLE 11.1 (continued)

P2.0	Member Data Source: P2.1 {I- Video_No} Source: Customer via a scanner I_Pay_Type Source: Customer via keyboard CC_No Source: Customer via mag reader	2.1 <u>Input the data for a rental</u> Generate (Rental_No, Date) Do Read {I_Video_No} from scanner Until no input Read (I_Pay_Type) from keyboard If (I_Pay_Type) = 1, then Read (CC_No) from mag reader Trigger 2.2 and send Rental and Member Data End	Rental and Member data Destination: P2.2	P2.2
P2.1	Rental and Member Data Source: P2.1 Approve_No Source: CC company Video Data Source: Video – External Database	2.2 <u>Calculate cost and process payment</u> Triggered by 2.1 and sent Rental and Member Data Perform Calculate-charge ({Video_No}, Charge) If Pay_Type = 1, then Perform CC approval (CC_No, Charge, Approve_Code) else trigger 2.3 [Note: Calculate-charge and CC-approval processes to be coded later] If approval denied, then End else trigger 2.3 & send Rental and Member Data End	CC number Destination: CC company Rental and Member data Destination: P2.3	P2.3
P2.2	Rental Data Source: P2.2	2.3 <u>Create rental and lines</u> Triggered by 2.2 and sent all rental data Insert Into Rental Values() Perform for j=1 to i Insert Into Line Values () Trigger 2.4 and send all rental data	Rental Data Destination: P2.4 Rental Data Destination: Rental Line Data Destination: Line	P2.4
P2.3	All Rental Data Source: P2.3 Title Data Source: Title, External File	2.4 <u>Print a receipt</u> Triggered by 2.3 and sent all rental data Retrieve the video titles from Title Perform Print-receipt End	Receipt Data Destination: Printer	P1.0

The maps can decompose the program into families of screens. A separate analyst can then program each family with minimal interaction with other analysts. Page action maps describe the layout and specify the actions and controls on the screens that hold the actions.

Page Navigation Maps

As noted in Chapter 8, Web sites and other dialog-driven systems can present a challenge for modeling. The team may wish to try using a page navigation map to provide a graphical framework for the system design of a Web site. Page navigation maps can display the navigation paths and data access requirements of a dialog-driven system. Dialog systems, especially Web designs, include simple text retrieval and, in some cases, processing or computational components. A Web site that displays course descriptions is a text retrieval system that requires little design documentation. A Web site that allows students to enroll in courses executes some complex functions in addition to text retrieval.

The page navigation map shows each page and the major links between pages. A page is the set of text and graphics displayed at any point in time by the Web browser. Menus and hypertext define many text retrieval systems. A menu displays a set of options. Clicking on a word, menu option, or box transfers the system to that location, possibly on the same page. The selection results in a control flow with no other data. Many modern Web sites contain a basic function menu across the top of every page that transfers from one branch of the site to another. Frequently, another menu along the left-hand side allows the user to select the section or function to access. The logic of these menus consists only of a simple list of display text and reference locations.

Web sites that perform some function or process logic require additional detail in the documentation. The page navigation map shows access to data stores using the same table symbols that appear on a DFD. Dotted arrows indicate the direction(s) of data flow. The general rules appear below.

1. Page. Each page of the site is represented by a rectangle. A page is the unit available from the browser to the viewer without clicking a hyperlink. Pages are numbered consecutively, 1, 2, . . . , n . An optional descriptor may follow the number, for example, P1. Home Page. The page name should correspond to the name used in the code: a form name, an html file name, or a display routine.
2. Links. Solid lines indicate **links**. They show the main links from the current page to other pages. All links are bi-directional unless the back function is disabled for a particular link.
3. Menus. The letter M followed by a number inside the page box indicates a **menu**. The menu numbers should be unique across the entire diagram. Menus with links to other pages receive labels that start with an M followed by a number (e.g., M1, M2, etc.). The menu number may be followed by an optional descriptor (e.g., M1. Main Menu). Frequently a site has one or several menus that appear on a number of pages. The same menu (using the same number and name) can appear on several pages.

4. Processes. A page may contain one or more processes on sites that allow the user to initiate some action, such as enter data, download a data set, transform data, and retrieve from or write to a database. Normally the process is triggered by the user clicking on an action button such as “Submit,” “Place in Shopping Cart,” or “Compute.” An action button is represented by the letter *A* followed by a number—A1, A2, and so on. A process may consist of one or more modules in accord with the rules for good module design.

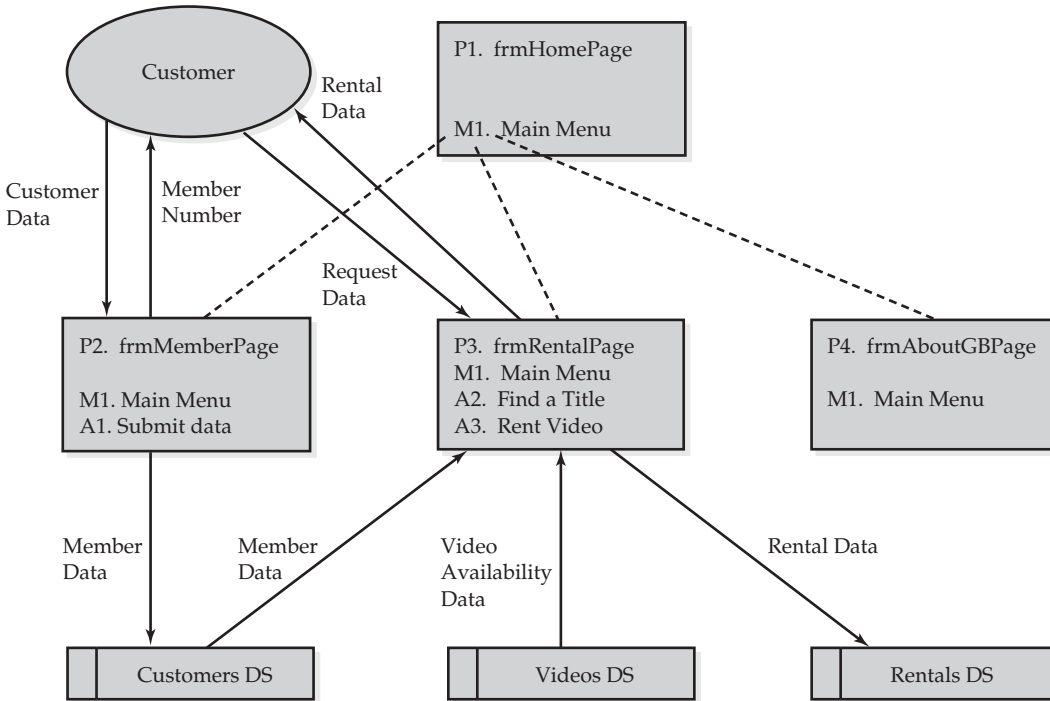
For complex processes, the analyst can create a DFD, structure chart, or FHD to represent the details of the process. Use DFD label conventions for the modules starting from the process label. Procedures may receive data from external sources, send data to external sinks, and store data in or retrieve data from a data store.

5. Data store. A procedure and only a procedure may interact with a data store to store or retrieve data. Use the DFD symbol for a data store. A graphical data model or data schema defines the data stores as discussed in the previous section.
6. Data flows. Flows show data transfers from or to externals, between modules, and to or from data stores. Use DFD rules and symbols, for example, all flows must have a process (i.e., a page with an action button) at one end, and the flows actually go to and from the process initiated by the action button. (*Note:* A user click on a hyperlink or menu represents a control flow not a data flow. A hyperlink click is not shown as a data flow from the user to the page. The control flow may initiate a data flow, for example, cause a procedure to retrieve data from a data store.)
7. Externals. Externals, for example, the site user or a credit card processor, are represented by the ellipse symbol.

An illustrative page navigation map for a Web-based GB Video rental system appears in Figure 11.6. To find out more about GB Video, the customer may click “About GB” on the main menu and view a page that tells about GB Video and gives branch locations and telephone numbers. Most customers probably will want to look at the availability of and/or rent videos. To make a rental, the customer must be a member. If the customer is not a member, he or she can become one by clicking “Become a Member” on the main menu. On the New Member screen, the customer fills in the customer data (see Figures 11.2 and 11.3) and presses a Submit button. The system creates a customer record and displays the customer number on the screen. The text asks the customer to print the screen for a written record of the customer number.

To rent a video, the customer clicks “Rent a Video” on the main menu. Note that the main menu (M1) appears on every page. On this page (P2), an index of videos appears. The customer can enter a member number, date, and the title of a desired video and click the Find button (A2). Or the customer can use the index to find a video. For each title on the selected date, the screen shows either “Unavailable, try again later” or “Available.” When the customer finds a desired video, the customer clicks the Rent button (A3). The customer selects either the pick up option or the delivery option. The system charges the credit card in the

FIGURE 11.6 Page Navigation Map for the GB Web-based Rental System



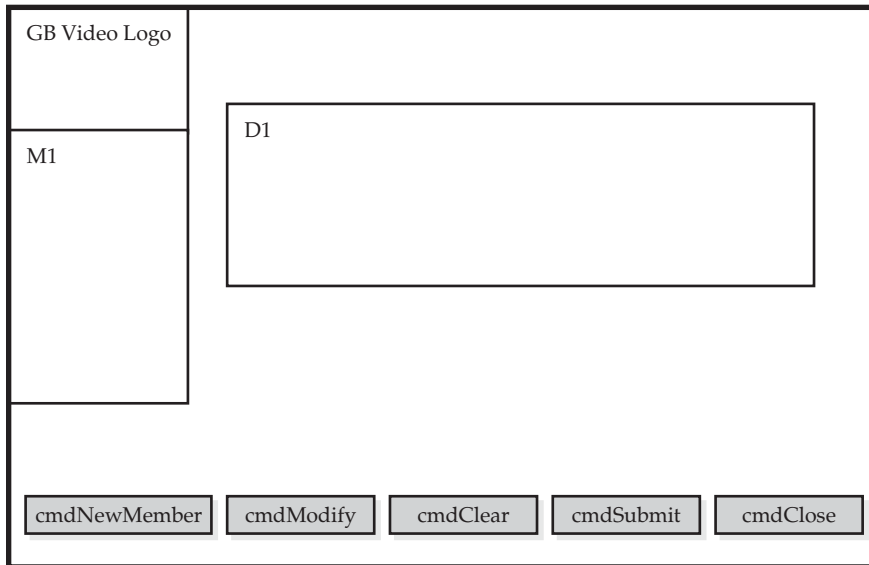
customer’s member record. The system displays the Rental data on the screen and invites the customer to print a hard copy. As the system is designed in Figure 11.6, each rental is for one video, but the customer can make multiple rentals if desired. Return and overdue functions are handled by the non-Web-based processes described earlier in the chapter.

Combinations of design tools may provide the best graphical process model for a site. For processes in the page map that contain more than several modules, a page action map, program structure chart, or physical DFD for the process (or one for the entire site) can add clarity and reduce ambiguity. A data schema model defines and clarifies the data store structure. Metadata further describes the objects on a Web page map in the same fashion as other graphical process models.

Page Action Maps

A page action map describes screen layout and actions and thus can complement the page navigation map that displays pages, data flows, and screen actions. The page action map provides a sense of the layout or the code specifications for actions, displays, or links on each page. Typical pages on an interactive system contain some of the following components:

- Pictures that display static graphics such as a company logo.
- Display fields that show the contents of some attribute or file.

FIGURE 11.7 Page Action Map for the GB Video Member Screen

- Links that connect to different locations in the system without passing or storing information.
- Controls that contain the code to execute some code or logic. Controls include buttons, action tabs, and entry fields.
- Events that trigger control action. Typical events include “click,” “double-click,” and “get focus.”

An action map indicates where these components should go and what they should do. Legacy mainframe systems documented these features with detailed character charts that specified exactly where each character, field, or symbol would be displayed. Modern GUI-developed systems are seldom that precise.

The page action map in Figure 11.7 shows an illustration for page P2, frm-Member Page from the page navigation map in Figure 11.6. The layout of the action diagram corresponds to the way that a designer might lay out the page. The GB logo goes in the upper left corner. Menu M1 appears below the logo. D1 is a data entry area in which the customer enters name, address, and other data. The rectangles at the bottom stand for action buttons that cause the following actions when the user clicks on a button:

- cmdNewMember. Set up D1 to accept the member data from the user.
- cmdModify. Allow the user to modify the member data stored in the system.
- cmdClear. Clear all existing data from D1.
- cmdSubmit. Send the data in D1 for processing.
- cmdClose. End the actions on the member page.

FIGURE 11.8 Metadata for frmMemberPage

M1 contains links to frmHomePage, frmMemberPage, frmReservationPage, frmAboutGBVideoPage

P2. frmModify

GBVideoLogo links to the image located at C:/GBVideo/Images/Logo

D1 contains individual text boxes linked to records from table Customer. Initial display should have Customer locked. Display should locate record matches for:

Cust_No

Tel_No

Partial letter entries for L_Name

Boxes display:

Cust_No. Text = Customer Number.

F_Name: dropdown. Text = First Name.

L_Name: dropdown. Text = Last Name.

Ads1. Text = Address.

Ads2. Text = Address.

City. Text = City.

State. Text = State.

Zip. Text = ZIP Code.

Tel_No. Text = Telephone Number.

CC_No. Text = Credit Card Number.

Expire. Text = Card Expiration Date.

cmdNewMember is a command button. Display regular. Status active. Text = New Member.

(on click)

Present a blank customer record in D1 with an auto-generated Cust No for user entry. Values should not be committed to the Customer table unless the Submit button is activated.

Set display of cmdClear and cmdSubmit to regular.

Set display of Modify to shadow.

Set status of cmdClear and cmdSubmit to active.

Set status of cmdModify to not active.

cmdModify is a command button. Display regular. Status active. Text = Modify.

(on click)

Unlock the display D1 for modification of all attributes except Cust_No. Values should not be committed unless the Submit button is activated.

Set display of cmdNewMember to shadow.

Set display of cmdClear and cmdSubmit to regular.

Set status of cmdNewMember to not active.

Set status of cmdClear and cmdSubmit to active.

cmdClear is a command button. Display shadow. Status not active. Text = Clear.
(on click)

Roll back any changes in D1.
Display current record from Customer.

cmdSubmit is a command button. Display shadow. Status not active. Text = Submit.
(on click)

Commit the values in D1 to table Customer.
Display current value of Customer.

cmdClose is a command button. Display regular. Status active. Text = Clear and Close.

(on click)
Close frmModify.

frmModify

(on open)

Display blank values in D1
Set status of cmdNewMember, cmdModify, cmdHome to active.
Set status of cmdClear, cmdSubmit to not active.
Set display of cmdNewMember, cmdModify, cmdHome to regular.
Set display of cmdClear, cmdSubmit to shadow.

(on close)

Roll back values in D1.

Page 2

Each of the structures is described in more detail in the metadata. The labels on the diagram and the descriptions use the names for forms and controls that will appear in the actual code.

In some cases the design team may take a prototyping view of page design and create the page before providing specifications for it. In those cases, the design documentation can consist of copies of the screens the team built with labels for each of the structures on the screens and corresponding descriptions in the metadata for each label.

Page Navigation and Action Map Metadata

As noted previously, metadata should communicate with a competent programmer who is familiar with the language in which the system is written. The documentation need not spell out *how* to create a control or display; rather the documentation indicates *what* the programmer should do. The metadata can consist of an abbreviated set of instructions for the page, written in terms that fit the language. Figure 11.8 illustrates possible metadata for page P2 in Figure 11.7.

DATA WAREHOUSE DESIGN

A **data warehouse** is the name for an information system that contains archived operational and related data used to make organizational decisions. Data warehouses support ad hoc queries and “what-if” decision analysis using data to identify problems or discover opportunities in time to take advantage of the information. The users query the data to gain insight for strategic and tactical decisions. Because data warehouse updates typically occur in batch mode, normalization offers little advantage and massive indexes are sustainable.

Dimensional Models

Data warehouse designs follow a **dimensional model** rather than a relational model. The dimensional modeling principle derives from work done at about the same time as work on relational databases. Kimball and others, 1998, and Inmon and Inmon, 2002, expanded and refined the original work and have pioneered modern data warehouse thinking. Dimensional models strive to maximize user understanding and ease of retrieval. The typical dimensional structure, a **data mart**, is designed around a **central fact table** that contains numeric values for analysis and dimension tables for keeping track of properties that determine categories and groupings. The head of GB Video might want a system that will allow him to examine the rental patterns of members so that GB can create more attractive packages and better manage video purchases and inventory levels. A data mart provides a tool to address this request.

Figure 11.9 shows the diagram for a data mart derived from the data model used by the GB Video rental system. The data in the data mart:

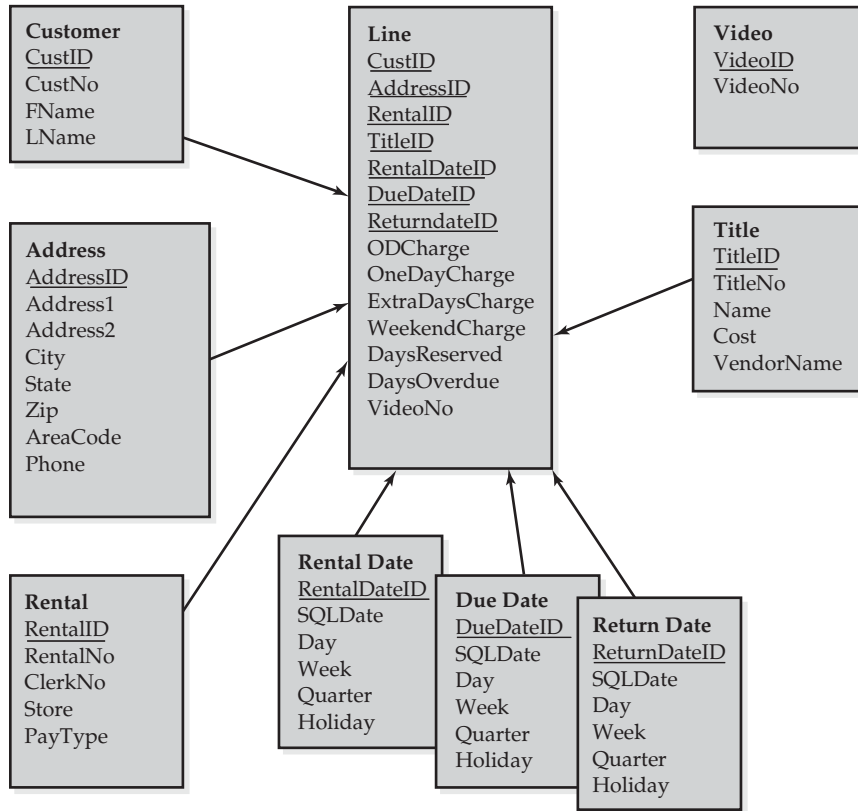
- Come from the everyday processing of video rentals and returns; and
- Support managers who analyze the data to learn about the rental behavior of customers.

The fact table “Line” is generated from the table with the same name in the GB Video rental system. The fact table contains “facts” that describe properties of each rental line and foreign keys that connect the fact records to records in associated “dimension” tables. Dimension models can be implemented directly in relational database engines and queried by SQL statements. More specialized models are implemented in multidimensional data cubes and retrieve data with spreadsheetlike queries.

The rules for converting the conceptual data model diagram for the GB rental proposed system shown in Chapter 8 into a data mart proceeds as follows:

1. *Select an associative entity to define the central fact table for the mart.* If there is more than one associative entity in the ER diagram, the system probably will generate more than one potential data mart. The records in the table that correspond to the associative entity define the *grain* of the fact table. The designer should explicitly describe what that grain is in operational terms. In this case the grain of the mart is, “An individual video rental line describing a specific video rented as part of a specific rental.”

FIGURE 11.9
Data Mart for
GB Video
Rentals



2. Replace all primary keys in data mart tables with artificially generated surrogate keys. Using the original primary keys raises potential problems because the data mart may span a longer time period than the operating database. For example, GB Video might reuse rental numbers each year without causing any confusion in the operational database. If the data mart contains data from several years, duplications will occur in the original primary keys. For this reason, a good data mart replaces the operational primary keys. Once the **surrogate keys** have been inserted into the tables, the original primary keys become redundant. However, the team may wish to retain the primary keys in case the need arises to research the origin of some entries after the mart is in place.

Surrogate keys are generated in a surrogate key index table for each dimension table in the final design. This table has three columns: the original primary key, the automatically generated key, and the permanent surrogate key. The automatically generated key column allows an automatic numbering function to continue generating keys as processing continues and rows are added to the dimension tables. These numbers will not be duplicates. If two different primary key values in the transaction system refer to the same thing, the third column can be adjusted to preserve the same surrogate key. If primary key values are reused over time, the designer may have to manage them separately.

3. *Create dimension tables from all the entities that relate to the central fact table by promoting foreign keys into the fact table.* In a relational database, a foreign key in the Rental table expresses the relationship connecting records in the Customer table to individual Rental occurrences. In a data mart, adding the foreign key for customer for each rental into the Line table maintains the information content and makes for a simpler query structure. This action denormalizes the table, that is, it creates a second normal form violation in the Lines table; however, it simplifies the structure and operation of the data mart.
4. *Promote dates to the fact table and create a date dimension to replace the actual date value.* The date dimension should contain the date from the operational file but also should contain properties of the date that are important for understanding. Retrospective analysis often requires grouping values by data properties that are difficult to derive from the SQL data stored in the transaction system. Questions such as “How many sales were made on Mondays?” or “Which holiday had the most video rentals?” require a complicated algorithm to compute date information from the standard date representation. In a data mart, date attributes of interest are generally precalculated.

When the data contain several dates, as in the GB Video example, the fact table should contain a surrogate key for each date. The actual database should implement these several copies of the date dimension as views or virtual tables. This action both saves space and makes the generation of date attributes more consistent.

5. *Identify the facts in the fact table.* Facts are attributes that describe the records in the fact table. The most significant facts are numeric and additive. Additive facts are those that can be added up across records and still have meaning. In sales marts like the GB Video mart, the most common facts are amount and quantity. Most order systems can record price, quantity, and amount. Price, for example, is not additive, but quantity and amount are. In the GB Data Mart example, the overdue charge is an additive fact. One-day fee, extra days, and weekend are not additive. The mart designer converted these into ODCharge, OneDayCost, ExtraDayCost, and WeekendCost and moved them from the Video table into the fact table. The designer also added DaysReserved and DaysOverdue by calculating the duration of the rental from the dates in the system. The new attributes are additive facts; they precalculate useful information and make it easier to derive price, if necessary.

The promotion of pricing information from the Video table into the Line table also solves another problem. Prices for videos often change as they get older. Hot new titles command a higher fee than old classics. Entries in dimension tables do not change over time. Values in the fact table represent the actual costs as they change over time.

6. *Identify the attributes in the dimension table.* Attributes are properties that are relevant to the purpose of the mart. These attributes are usually text or categorical variables. Occasionally numeric variables appear as attributes in dimension tables when there is no expectation of doing arithmetic with the values. Attributes that are irrelevant or that might provide a security risk should not be included. For that reason, credit card number and expiration date do not appear in the mart.

In addition to keeping existing attributes, the designer should replace codes with useful values. That is why the vendor number has been replaced with VendorName in the Title table. The Store attribute has been added to the rental table so that reports can be generated by store if needed.

7. *Manage slowly changing attributes.* A slowly changing attribute is an attribute in a dimension that will change over time but much less frequently than values in the fact table do. If these values change frequently then they should be included as attributes in the fact table, in essentially the same way that the OneDayCost, ExtraDayCost, and WeekendCost were included in the fact table rather than the Video table. Customer address is another situation. Address may change, but it would be a waste of space to include it in the fact table. There are three rational approaches called Type 1, Type 2, and Type 3 approaches.

Type 1: Keep only the current values.

Type 2: Create another table to keep each change.

Type 3: Add an attribute to the dimension table to keep the previous value.

The choice depends on the purpose of the mart. In the GB Video case it would be useful to understand where sales have come from based on where the customer lived at the time of the rental. The design discussed above treats address as a Type 2 variable, which is why there is a separate address table.

8. *Remove unnecessary dimensions.* Dimensions that are left with only a single attribute serve no useful purpose and take space and processing power in the fact table. In this example, the Video dimension that originally contained information about the individual tape has only the VideoNo attribute in it. Since that attribute has no value to the purpose of the mart, the designer has chosen to omit that dimension.

The data mart for GB Video in Figure 11.9 contains essentially the same information as the CDM for GB Video but follows a different structure tailored to its purpose. The dimensional model is heavily denormalized to improve retrieval performance.

Data Warehouse Metadata

Metadata on the GB Video data mart appear in Figure 11.10 using the format from Kimball (1998).

The Extraction-Transform-Load Process

Once a data mart is designed, the team must populate it with data and must establish an ongoing process to maintain the data in correct form. This process is called the **extraction-transform-load (ETL)** process. Extraction refers to the processing necessary to extract data from the source systems into a work area where it can be prepared for the final data mart. The transform process includes any edit or cleansing routines as well as the structural transformations (generating surrogate keys, creating additive facts, splitting compound attributes) necessary to generate data in the new form. Finally, the load process involves inserting the appropriate data into the warehouse for use.

A data flow for a typical ETL process is shown in Figure 11.11. As indicated by the table references in the diagram, much of the metadata for the processes

FIGURE 11.10 Metadata for the GB Video Data Mart

1. A description of data sources.

Source	Organization Owner	IT Owner	Platform	Location	Data Source Description
GB Video Rental System	Operations	IT Director	Corporate Server	Home Office	GB system for transaction processing.

2. A description of the source to target extraction tables used to create the initial source for editing. These working tables are similar to the ultimate presentation tables in the data mart. They contain data in the same format as the source system.

Target Table	Target Column	Data Type	Len	Target Column Description	Source System	Source Table / File	Source Col / Field	Data Txform Notes
Customer	CustID	Integ	16					Surrogate
Customer	CustNo	Num	8	Old primary key	Rental	Customer	Cust_No	
Customer	FName	Text	20	First Name	Rental	Customer	F_Name	
Customer	LName	Text	30	Last Name	Rental	Customer	F_Name	
Address	AddressID	Integ	16					Surrogate
Address	CustNo	Num	8	Customer primary key	Rental	Customer	Cust_No	Delete after surrogate key
Address	Address1	Text	40	Address line 1	Rental	Customer	Ads1	
Address	Address2		40	Address line 1	Rental	Customer	Ads2	
Address	City		25	City	Rental	Customer	City	
Address	State		25	State	Rental	Customer	State	
Address	Zip		10	ZIP	Rental	Customer	ZIP	
Address	Phone	Integ	10	Customer Phone	Rental	Customer	Tel_No	
Rental	RentalID	Integ	16					Surrogate
Rental	RentalNo		16	ID for rental receipt	Rental	Rental	Rental_No	

Rental	ClerkNo		16	Clerk employee number	Rental	Rental	Clerk_No	
Rental	Store		40	Store Name	External			Look Up
Rental	PayType		25	Method of payment	Rental	Line	Pay_Type	
Title	TitleID	Integ	16					Surrogate
Title	TitleNo	Text	10	Video Primary Key	Purchasing	Title	Title_No	
Title	Title		70	Video Title Name	Purchasing	Title	Title	
Title	Cost	Curr	10	Video cost	Purchasing	Title	Cost	
Title	VendorName	Text	50	Supplier for video titls	Purchasing	Vendor	Vendor-Name	
Line	CustID	Integ	16	Customer SK				Surrogate
Line	AddressID	Integ	16	Address SK				Surrogate
Line	RentalID	Integ	16	Rental SK				Surrogate
Line	TitleID	Integ	16	Title SK				Surrogate
Line	CustNo	Text	8	Customer Original Foreignoriginal Key	Rental	Rental	Cust_No	Delete after Replace with surrogate
Line	RentalNo			Rental original foreign key	Rental	Line	Rental_No	Delete after Replace with surrogate
Line	TitleNo			Title original foreign key	Purchasing	Video	Title_No	Delete after Replace with surrogate
Line	ODCharge	Curr	12	Charge for overdue returns	Rental	Line	OD_Charge	Delete after transform process

Line	OneDayFee	Curr	12	Daily fee for the first day	Rental	Comvideo	One_Day_Fee	Delete after transform process
Line	ExtraDayFee	Curr	12	Daily fee for each day after the first	Rental	Video Comvideo	Extra_Days	Delete after transform process
Line	WeekendFee	Curr	12	Premium charged for weekend days	Rental	Comvideo	Weekend	Delete after transform process
Line	OneDayCharge	Curr	12	One-day charge for video				Calculate
Line	ExtraDaysCharge	Curr	12	Charge for days after the first				Calculate
Line	WeekendCharge	Curr	12	Premium for weekend use				Calculate
Line	DaysReserved	Integ	8	Number of days reserved				Calculate
Line	DaysOverdue	Integ	8	Number of days overdue				Calculate
Line	RentalDateID	Integ	16	Date surrogate for rental date				Surrogate
Line	DueDateID	Integ	16	Date surrogate for due date				Surrogate
Line	ReturnDateID	Integ	16	Date surrogate for return date				Surrogate
Line	SQLRentalDate	Date	20	Rental date	Rental	Rental	Date	Replace with surrogate
Line	SQLDueDate	Date	20	Due date	Rental	Line	Due_Date	Replace with surrogate
Line	SQLReturnDate	Date	20	Returned date	Rental	Line	Return_Date	Replace with surrogate
Date	DateID	Integ	16					Surrogate

Date	SQLDate	Date	20	SQL date value				Calculate
Date	Year	Int	4	Year				Calculate
Date	Day	Text	1	Day of week				Calculate
Date	Week	Integ	4	Week of year				Calculate
Date	Quarter	Integ	4	Quarter of year				Calculate
Date	Holiday	Text	30	Holiday Name				Calculate

3. A description of the transformation analysis needed to clean or reformat the source data.

Derived Fact Name	Derived Fact Description	Type	Agg Rule	Formula	Constraints	Transformations
ExtraDaysCharge	Charge for additional reserved days			(Extra_Days)* (DaysReserved - 1)		
WeekendCharge	Additional charge for weekend use			Weekend*(If reserved over weekend)		
DaysReserved	Total days reserved			Due_Date - Rental_Date		
DaysOverdue	Days kept in addition to reserved days			Return_Date - Due_Date		

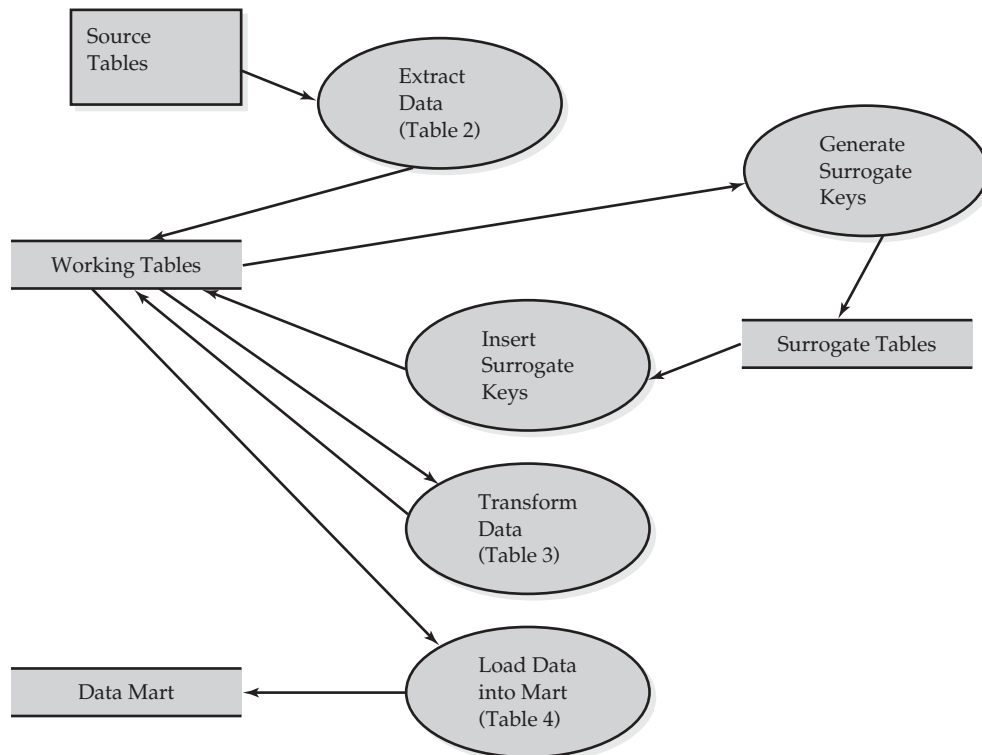
4. A description of the ultimate data mart tables.

Table Name	Column Name	Data Type	Len	Nulls?	Column Description	PK	PK Order	FK
Customer	CustID	Integ	16	N		Y	1	
Customer	CustNo	Num	8		Old primary key			
Customer	FName	Text	20	N	First Name			
Customer	LName	Text	30	N	Last Name			
Address	AddressID	Integ	16	N		Y	1	
Address	Address1	Text	40		Address line 1			
Address	Address2		40		Address line 1			
Address	City		25		City			

Address	State		25		State			
Address	Zip		10		ZIP			
Address	AreaCode	Integ	5		Customer Area Code			
Address	Phone	Integ	7		Customer Phone			
Title	TitleID	Integ	16	N		Y	1	
Title	TitleNo	Text	10		Video ID from Rental System			
Title	Name		70		Video Title			
Title	Cost	Curr	10		Video cost			
Title	VendorName	Text	50		Supplier for video title			
Line	CustID	Integ	16	N	Surrogate for Customer	Y	1	Y
Line	AddressID	Integ	16	N	Surrogate for Address	Y	2	Y
Line	RentalID	Integ	16	N	Surrogate for Rental	Y	3	Y
Line	TitleID	Integ	16	N	Surrogate for Title	Y	4	Y
Line	RentalDateID	Integ	16	N	Date surrogate for rental date	Y	5	Y
Line	DueDateID	Integ	16	N	Date surrogate for due date	Y	6	Y
Line	ReturnDateID	Integ	16	N	Date surrogate for return date	Y	7	Y
Line	DateID	Integ	16	N	Date surrogate for rental date	Y	11	
Line	ODCharge	Curr	12		Overdue Charges			
Line	OneDayCharge	Curr	12		One day charge			
Line	ExtraDayCharge	Curr	12		Charge for an extra day			
Line	WeekendCharge	Curr	12		Charge for weekend rental			
Line	DaysReserved	Integ	8		Number of days originally reserved			
Line	DaysOverdue	Integ	8		Days overdue when returned			
Rental/ Due/ Return Date	SQLDate	Date	20	N				

Rental/ Due/ Return Date	Year	Int	4	N	Year			
Rental/ Due/ Return Date	Day	Text	1	N	Day of week			
Rental/ Due/ Return Date	Week	Integ	4	N	Week of year			
Rental/ Due/ Return Date	Quarter	Integ	4	N	Quarter of year			
Rental/ Due/ Return Date	Holiday	Integ	4		Holiday Name			

FIGURE 11.11 ETL Data Flow for a Data Mart



is contained in the tables. The diagram describes the operational sequence necessary to actually load the data mart. The transformations described here are quite simple and do not include any attribute reformatting or data editing that may be needed. In practice, data preparation might be quite extensive and lead to an expansion of the Transform Data process into a complex subprocess.

Summary

System design focuses on program architecture and the interaction of programs with the physical and organizational infrastructure in which the programs will operate. Unlike the conceptual solution specifications discussed in Chapter 8, system design addresses conceptual, logical, and physical issues within the structure of the specific technologies and organizations that the system plans to employ. The selection of the technologies for a system is an art and often depends on personal preferences and/or organizational policies. For example, some people prefer procedural languages and design while other believe that object-oriented design works better for many applications.

The goal of system design is to create detailed specifications that will enable programmers to write and maintain correct and complete code for the proposed system. Good design documentation should communicate to an experienced developer the information that he or she needs to write and/or maintain the code. The documentation specifies fully how the program works in an easy-to-follow format. The content of the documentation depends both on the system content—data, process, and physical and organizational infrastructure, and on the design approach selected by the team. Design documentation matches the conceptual specifications for the program to the logical and physical specifics of the program environment. Good design and documentation work together to simplify the coding and maintenance process.

System design utilizes the most detailed models of the systems solution process. Because system design takes place at a detailed level in a specific technological and organizational environment, the appropriate models depend on the environment. For example, such process models as program structure charts may work well for applications implemented in procedural languages. Other models may better represent such applications as Web sites and data warehouses or such approaches as object-oriented design.

During system design, the team reviews each of the content areas and adds content or detail to arrive at a complete design that when operational will solve the problem posed by the client. System design activities may include the preparation of specifications and models for:

- Physical infrastructure. Defines the hardware and software as appropriate to facilitate performing the activities of the new system.
- Organizational infrastructure. Modifies the organization as needed to match the functioning of the new system.
- Data. Converts the conceptual data model into such a graphical data design as a relational schema with the associated metadata.
- Process. Expands the conceptual process model to specify the detailed system logic with graphical models and metadata.

The appropriate documentation for a system depends on the nature or function of the system. Design documentation for process-driven systems may include a DFD or program structure type process model and metadata. Most Web and visual applications are dialog driven. Such user-activated events as key strokes and mouse clicks determine the sequence of actions executed by the program. Design documentation may include page navigation maps, page action maps, and detail process models for screens that execute processes other than navigation.

Most designs involve modules—a related group of logic or program statements. Program structure charts and physical DFDs directly show modules. Page navigation maps may contain modules associated with actions for a page. In OOD, modules appear as operations and their related messages. Good design strives to create modules with high cohesion and limited coupling, primarily data coupling. The logic of modules is specified in terms of triggers, input and output, and code or pseudocode for the processes in the module. Pseudocode provides a standardized framework for describing process logic that a programmer can translate into program code.

Data warehouses provide read-oriented database structures for such functions as reporting and strategic data exploration. Documentation for a warehouse may include source data, source to target extraction tables, data mart diagrams and tables, transformations to clean and reformat data, and ELT process diagrams.

Key Terms

central fact table, 408	extraction-transform-load (ETL), 411	page navigation map, 399
cohesion, 390	link, 402	physical data flow diagrams, 389
coupling, 390	maintainability, 376	program structure chart (PSC), 387
data-driven, 387	menu, 402	pseudocode, 392
data mart, 408	module, 387	security, 378
data types, 383	module cohesion, 376	structured code, 376
data warehouse, 408	module coupling, 376	surrogate key, 409
dialog-driven, 387	network, 378	TIPOT chart, 399
dimensional model, 408	page action map, 399	

Review Questions

Answer the following questions regarding these topics.

- Design documentation.
 - Who is the audience for good design documentation?
 - How does design documentation differ with different development environments or languages?
- Design components.
 - What should design provide specifications for?
 - What are the goals of good design?
- Good design.
 - What are some design rules that lead to code that is easy to maintain?
 - How does design support allocation of workload to more than one person?
 - What are the critical organization roles that a design team should identify?

4. Data.
 - a. What are the components of a complete data model?
 - b. How does a design model for a database differ from an analysis model (CDM)?
 - c. What metadata should be supplied for data in the design?
5. Process.
 - a. What are program structure charts, physical data flow diagrams, object schema, and page navigation maps?
 - b. What is the difference between process-driven, data-driven, and dialog-driven design? Why might the documentation requirements be different?
6. Modules.
 - a. What are coupling and cohesion?
 - b. What are guidelines for good structured modules?
7. Pseudocode.
 - a. What is pseudocode?
 - b. What are the basic functions described by pseudocode statements?
 - c. What is a TIPOT table?
 - d. What is a stubbed procedure module?
8. Dialog-driven systems.
 - a. What does a page navigation map show?
 - b. What does a page action diagram contain?
9. Data warehouses.
 - a. How does a data warehouse differ from a process-driven system?
 - b. How does a dimensional model differ from an ERD?
 - c. What is the ETL process?

Critical Thinking Exercises

Individual Exercises

1. For the logic in Process 3.0
 - a. Prepare the metadata.
 - b. Show the metadata in a TIPOT table.
2. Complete design documentation for GB Video
 - a. Using process-driven forms.
 - b. Using dialog-driven forms.

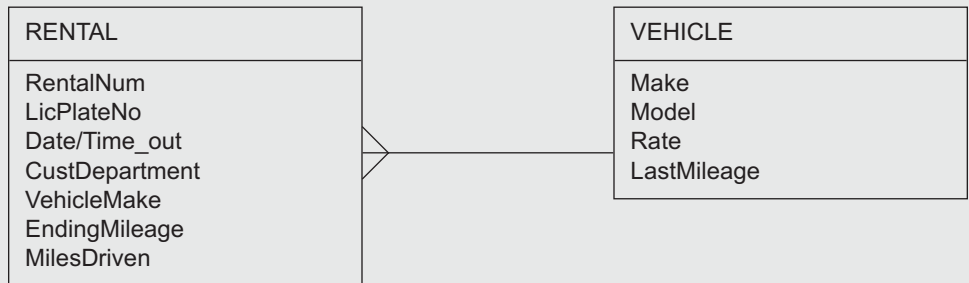
Group Exercises

1. Use the Motor Vehicle Pool case in Chapter Three for the following questions.
 - a. Produce a program structure chart. Use correct numbering and naming conventions. Indicate any data that flow among modules. Prepare the relational schema for the design.
 - b. Provide complete design documentation for the Return a Vehicle process modules in expanded IPO format. Include appropriate triggers, input, process, output, and triggered events. Write a pseudocode (or similar) descriptions of the process logic.

2. The administration for the Motor Vehicle Pool wants to create a data warehouse to track vehicle use. Questions include
- How far has each vehicle been driven each month?
 - What are the number of rentals, total miles, and total charges for each department each month?
 - What is the total mileage on each vehicle?

Data warehousing specialists have designed the following structure for the data mart.

- a. Draw a data flow diagram for the update process (ETL) for the data mart.



- b. Provide complete design documentation for the Load Data into Rental process. Include appropriate input, process, and output. Write a pseudocode (or similar) description of the process logic.
- c. Write pseudocode (or SQL) to retrieve “Total miles rented by each Customer Department each month” from the data mart.
3. OSU has decided to move the reservation part of the rental system to the Web. They have indicated that they want a home screen for the rental program that includes a description of the department and the mission of the program. From that page they want to be able to select a Vehicles Owned sheet that describes the types of vehicles available and their rate. They also want a Rent Vehicles screen that allows customers to reserve a vehicle for rental. The Vehicles Owned link brings up a MS Word document called Vehicles.doc stored in the folder OSURental on the OSU administrative server. The Rent Vehicles link goes to a screen that allows users to browse and reserve vehicles. The screen should operate as follows. The customer enters his or her Name, Telephone, Address, Department, and departmental Account_no. Once the account number has been verified (customer is on a list of people authorized to rent on the account) the screen displays a box for the desired Date/Time_in and Date/Time_out. When this information is entered, the system displays a list of all vehicles available (LicPlateNo, Make, Model). A vehicle is available if there is no rental record for that vehicle that overlaps with the rental time requested. The customer clicks on a vehicle and the system records a rental entry for that vehicle and customer.
- a. Draw a Web map to track the navigation through the system.
- b. Draw a screen layout for the Rent Vehicles screen. Include any trigger buttons you need.
- c. Provide complete design documentation for the Rent Vehicle process. Include appropriate input, process, and output. Write a pseudocode (or similar) description of the process logic.

References

Demarco, Tom. *Structured Analysis and System Specifications*. Upper Saddle River, NJ: Prentice Hall, 1978.

Inmon, W. H. *Building the Data Warehouse*. 3rd ed. New York: John Wiley & Sons, 2002.

Kimball, Ralph; Laura Reeves; Margy Ross; and Warren Thornthwaite. *The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing, and Deploying Data Warehouse*. New York: John Wiley & Sons, 1998.

Yourdon, Edward; and Larry Constantine. *Structured Design: Fundamentals of a Discipline of Computer Program and Design*. Englewood Cliffs, NJ: Yourdon Press, 1986.

Chapter Twelve

Proof of Concept

Chapter outline

Introduction

Types of POC Models

Package POC Models

Prototype POC Models

Using a POC Model

Evaluating Operational Feasibility

Examining Functionality

Examining Usability

*Evaluating Design Parameters and
Compatibility*

Prototype-Based Design

Building a Prototype

Choosing a Focus

Evolutionary Model Issues

Model Content Issues

Making Initial Design Decisions

Generating Code

Schedules and Assignments

Coding and Design Specifications

Control Flow

Data Model

Naming

Data CURD Operations

Logic

Effective Coding

A GB Video Prototype

Creating the Tables

Creating the Table Design

Creating Table Relationships

Populating the Tables

Coding the GB Prototype

The Switchboard

frm20CreateOrUpdateMemberData

*frm21InputCustomerAndVideoDataFor-
ARental*

*frm22ReceiveCashCheckOrPostACC-
Transaction*

Module23CreateRentalAndLines

frm24PrintARceipt

Summary

Key Terms

Review Questions

Critical Thinking Exercises

Individual Exercises

Group Exercises

INTRODUCTION

A **proof of concept (POC) model** provides a representation of the final application software for a project that allows the team and client to directly observe some characteristics of the system. POC models exist in operational and static versions. An **operational proof of concept model** converts inputs into outputs, stores and retrieves data, and exercises some or all of the processes in the system. A **static POC model** illustrates the outputs that result from a predefined set of sample inputs. Sometimes an actual system produces the output that corresponds to each input; at other times an analyst generates the outputs. The model only demonstrates the relationship between the predefined inputs and the corresponding outputs. The client cannot suggest inputs and see the resulting outputs in real time.

Most of the analysis and design tools used in the earlier stages of the SDLC are not operational models. A DFD provides a nonoperational graphical representation of how a system transforms inputs to outputs. DFDs, while helpful tools, may contain ambiguity, errors, or omissions that remain undetected until the system begins to operate. Operational models provide a much higher confidence level that selected aspects of a proposed system do or do not operate as designed and/or as expected. Whenever possible, the team should use an operational POC model.

A proof of concept model may serve one or more of the following purposes:

- To verify operational feasibility.
- To improve or refine the system specifications.
- To evaluate compatibility with a physical infrastructure.
- To confirm system design parameters.

Operational feasibility assesses the impact of the solution on the way the organization wishes to operate. The team often uses a POC model to verify the operational feasibility of the proposed system—to demonstrate to the client and the project team how a system based on the design specifications actually will perform. The proposed system specifications process described in Chapter 8 focuses on determining the features for the new system. Operational feasibility evaluates how and whether or not these features work together to solve the client's problem. While operational models provide more confidence, static POC models can help the team to explore some aspects of operational feasibility.

A POC model can demonstrate the two basic components of operational feasibility: functionality and usability. To test **functionality**, the analyst or client can present typical external inputs and let a POC model transform the inputs, for example, update data stores, trigger processes, and/or produce external outputs. The team and the client observe the POC model to determine whether all of the desired features are included and work as intended. In addition to demonstrating function, many POC models also demonstrate the **usability** of a system—how easy or satisfying or “likable” the system appears to users. Usability becomes clear only with a physical representation of the system. For example, users want

the functions that they use together to appear in a group on an input display. Users also want clear, informative display labels and help displays in the language they use for their work. Users want easy ways to generate reports with good summaries and graphics. A proof of concept model gives the client, users, and team an opportunity to explore these usability issues.

A proof of concept model can help the team and client to refine and improve the system specifications. Observing system operation and seeing actual output often generates ideas that neither the clients nor the team thought of in advance. While clients often find developing specifications difficult in the abstract, they know what they want and what they like and dislike when they see it. Many times clients identify some of the essential requirements and mandatory features only after they interact with the POC model. The proof of concept model allows the team to identify these additional issues prior to a final procurement or build action.

The team can use an operational POC model to confirm some of the system design parameters if and when the POC representation contains the fully functional design structure of the final system. With these models, the team can do such things as test the database design, exercise import and export functions, or check the module call sequences. However, some proof of concept models stub parts of the logic and processing behind the input/output (I/O) interfaces that are most important to the client and users. These I/O focused models may not allow the team to confirm many of the design parameters for the final system.

In selected cases, an operational proof of concept model, particularly a POC version of a purchased system, will allow the team to evaluate how well the proposed system will interact with the organization's infrastructure. Purchased system vendors may provide trial systems that are a limited functionality version of the production system. Installing and operating the trial system can give the team a first reading on whether the existing infrastructure will interface with the proposed product. However, the trial system may not offer much insight on such system performance issues as response time under load.

The next part of the chapter discusses package and prototype POC models and the use of POC models. The final materials discuss building POC prototype models and illustrate the steps to build a GB Video POC model in Microsoft Access.

TYPES OF POC MODELS

Proof of concept models consist of two major categories, prototypes and packages. The team builds a **prototype** POC model; the team sets up and populates a data schema and writes the program code. The prototype contains a data schema either identical to the final system or a close approximation. The code for the prototype may simplify or even omit parts of the final system code as long as the prototype meets the basic POC criterion by demonstrating the transformation of key inputs to outputs for the client and the team.

When the team recommends that the client purchase a package, either the actual package or a demonstration version of it may serve as the POC model. Most vendors will provide a version of their product for demonstration purposes:

the actual package, a demonstration version, a Web version, or a set of screen shots. A **package POC model** may require substantial work by the team before it will operate correctly or at all. If no trial version of the package is available, the team may create a prototype to demonstrate the features of the package to the client.

In certain cases, the team may be forced by circumstances to use a simulated POC model. A **simulated POC model**, a static model, consists of actual or fabricated screen shots or forms for input and output that tell a consistent story. Using the design specifications for the system, the inputs should produce the corresponding outputs. With this approach, the team shows and explains the sample input and then shows and explains the output that is expected. This approach can show the client what to expect, but gives up the major advantages of operational models—the confidence that the system works as intended. When a package vendor provides actual screen shots, the team and client can increase their confidence that the system actually works as pictured by talking with other users of the package.

When the team fabricates screen shots for the simulated POC model and no operational version actually exists, the POC provides little confidence that the final system will work as expected. A prototype that represents the I/O dialog correctly but uses mostly stub modules and report generators is another variation of a simulated POC. These models accept realistic input and produce representative output but may not use any of the logic or data structure for the actual proposed system. The team should use a simulated POC model only as a last result when an operational POC model is not available or is beyond the limit of available team resources. The team may be unable to obtain a demo package or may not have the skills and time to build a prototype.

Package POC Models

When the recommended solution specifies a package program, the team frequently can demonstrate some version of the package as a POC model. The amount that the team can learn from a demonstration version depends on what the vendor will provide and on the complexity of installing the package. A vendor may offer the following:

- A fully functional system on a short-term trial basis. Some vendors will provide a full version of their package with a limit on use to demonstration purposes for a specified period of time.
- A limited function system available for trial use. The demonstration version may contain such restrictions as limits on file sizes and simultaneous user access to prevent the demonstration from being used as the final system.
- Remote (often Web) access to a demo package that may or may not be customizable or operational. Some vendors provide access only to a static demo that shows screen shots for a sample set of input and for output reports that correspond to the sample input. This type of nonoperational demo provides a less helpful POC than operational demos, but may be the only version available.

Demo versions of packages may include a set of sample data in the form of tables already populated with some data. Sometimes, but not always, the team can customize the demo by adding data specific to the client.

If the team decides to use a version of a package as the POC model, the team can review the following checklist:

1. **Cost.** Is a demo available for free or at nominal cost? If not, will the client pay for the cost of a demo? Some vendors charge for supplying a demo.
2. **Schedule.** Can the team obtain and install the demo on a schedule acceptable to the client? If it takes months or years to get a demo, the client may need to make a decision before the demo arrives. The team needs the demo well in advance of the decision review time in order to install, test, and become familiar with the demo prior to showing it to the client.
3. **Configuration.** Does the team understand how to set up the demo to demonstrate the features of interest to the client? Large packages come with many options and require extensive configuration, some of which may involve high levels of complexity. The vendor may offer a package preconfigured to represent a typical installation that may serve well as a demo.
4. **Infrastructure.** Does the team or client have access to the necessary hardware and software to run the demo? For example, the demo may require a UNIX operating system and an Oracle Server database management system (DBMS). The client may not have and be unwilling to get the resources to run the demo.
5. **Effort.** How much work and expertise are required to install and initialize the demo? Some packages require effort and expertise well beyond that available to most teams. Will the client or the vendor supply resources to help with the installation of the demo?

Although installing a package offers the best opportunity of evaluating all of the functionality of the proposed system, package installation and operating issues can raise serious problems for the team. Using a remote access demo that the vendor installs and runs will eliminate the installation problems for the team. While many remote demos give the team the same functionality and data choices as the full package, some remote demos may not allow the team to provide data and may not actually operate. When the team installs and runs the demo, the team normally can provide data and select the functions to demonstrate. However, installing all of the functionality of the actual package may take more time and effort than is available. The more complex the system, the more effort is required to install, populate, and customize the system. Small, simple packages present few problems, but the large, complex systems available from such vendors as SAP, Oracle, and PeopleSoft can take years to set up. In these cases, the team may actually prefer to use a noncustomized system, perhaps installed at some other location, as the POC model.

With both installed packages and remote demos, the team must make certain that it actually can operate the demo. Some demos contain so much functionality and offer so many choices that learning to use all the features may require special training and/or a lot of time. To reduce the learning problem, the team

should determine the most critical functions or issues for the client and focus on demonstrating them. A lot of practice and rehearsal prior to a client demonstration are essential. If the team encounters problems during a demonstration to the client, the client may form a negative opinion of both the team and the package.

Prototype POC Models

When the team builds a solution or when no demo version of a package system is available, the team may choose to build a prototype. The prototype may range from a full-featured initial version of the actual production system to a highly simplified version that only provides some input to output transformations. Prototype-based POC models of a proposed system can reduce some of the features, infrastructure, and installation problems of package POC models. The team can build a prototype with features that match exactly to the design specifications. The team can tailor the prototype to already available or planned software and hardware. Normally, the team configures and may implement the prototype in part or in full during the process of constructing it. To incur these benefits, the team must incur the costs, particularly the effort and time costs, of actually building the prototype. Building the prototype is a subproject for systems development inside the main project, and all the tools and principles in this book also can apply to the subproject.

Prototype designers face a number of choices, a main one of which relates to the choice of a throwaway or evolutionary model. The team can create and use a **throwaway prototype** to develop, refine, and demonstrate the system. The prototype provides a tool to clarify the design and operations of the proposed system before the client invests in an expensive and less flexible production model. A throwaway prototype probably will not contain some of the features of the production system. For example, the prototype may use Access; while for scale reasons, the final application must use a large, multiuser DBMS, such as Oracle, Informix, and MS SQL. The prototype may work for only one or several users while the production system will build on a packaged transaction processing module to support many users.

After the throwaway prototype has produced the desired learning and insights, the client builds or contracts for a production system with new code and schema. Because the throwaway prototype can represent the design specifications for the production system, good documentation is important. The documentation should focus on the features and specifications for the system and not on how the prototype does it. Although some or all of the ideas and features from the prototype are transferred to the production programs, the original prototype itself is, in effect, thrown away.

An **evolutionary prototype** evolves into the final **production programs**, or programs that are implemented in the organization to carry out activities for the client. To facilitate a smooth transition from prototype to production system, evolutionary prototypes often are built with the languages, tools, data models, and infrastructure used by the client's organization for production systems. In a number of field projects, especially in smaller, low-budget operations, the client may wish to use the prototype produced by the team for a POC model as the production system.

In many projects, a prototype that uses Microsoft Access or a similar tool may have adequate capacity and features to handle small applications. In these situations, the team builds an initial version of the production system and uses it as a POC model to get client and user input. The prototype, perhaps with minor changes, becomes the production program. Some organizations build evolutionary prototypes for a range of systems from small to very large, very complex ones.

Prototyping also can represent a model for planning and managing system development. The spiral model for system development assumes that the team will use prototyping as part of the development process. Later sections of this chapter discuss (1) prototyping as a development tool and (2) the process of building prototypes.

USING A POC MODEL

Once the team possesses a POC model, either a package or a prototype, the team can use the model to validate and refine the design. The team normally starts by using the prototype for just the team. After the team concludes that the POC is performing as expected, the team can involve the client and/or users. The team can use the POC to address the issues discussed earlier: to verify operational feasibility, to improve and refine the system specifications, to evaluate compatibility with a physical infrastructure, and to confirm system design parameters.

Evaluating Operational Feasibility

Many POC models focus primarily on operational feasibility. To evaluate operational feasibility, the team can execute typical use patterns for the system and observe what happens. Asking users to identify typical use patterns and participate in the walk-through can add realism to the exercise. The team can develop scripts that follow the typical use of the system. The scripts should exercise most of the common ways that the system will be used, both for input and for reporting. Or the team can let users try things out on their own. The script approach works with both prototypes and packages, and, with a substantial amount of creativity, may work with a simulated POC.

Running against a script requires the POC model to execute against data. The data for the POC exercises may differ from the data for testing evaluation (see Chapter 13), but the POC data at a minimum should allow the team to demonstrate realistically the functions of the system. Reports should contain at least several data rows, and screens should scroll through several records. The system should demonstrate complete transactions. POC data become more critical when the team lets users run the system. Users who find unrealistic or incorrect data may lose confidence in the new system.

Examining Functionality

A POC model session with a user or client should follow the typical use pattern script to demonstrate the basic functionality of the proposed system. Users want a demonstration of the system's functions in the context of their job. In a POC demonstration to users, the team may leave out less frequently used functions

in order to focus on the users' primary concerns. The demo tries to answer such questions as:

- Will the system do the functions I currently use to perform my job with less or the same effort as required now?
- Will the system perform functions I need that the current system does not support?

At the end of the script, users appreciate the opportunity to suggest actions, such as, "Show me how the system will do the following . . ." In general, the team asks users to hold their questions until the end. Many questions are answered more efficiently by running the script.

Examining Usability

Usability questions that the team can ask themselves and users during the walk-through of the POC include:

- Do the input formats follow a logical sequence? A good design will flow naturally along the most common use sequence. Functions normally performed together should appear in the same screen or input form if possible. The team wants to avoid having the users ping-pong back and forth between screens or formats.
- Can a typical user easily learn the system? Clear labels for everything can help. The system should provide help instructions when a user selects an invalid choice or needs something unusual.
- Does the system trap and correct when possible invalid data input entries? Common problems are alphabetical characters that are placed in numeric fields, negative values in fields that always are positive, blank or null values in required fields, and dates in incorrect formats. Trapped error routines should gracefully prompt users for corrections and not lock up the system or e-mail a supervisor.
- Are the screens or formats both efficient and attractive? Putting too much on one screen or not aligning appropriate fields can make a screen unpleasant to use. Strident colors, unnecessary motion, or frequent flashes can distract and irritate users.
- Is there a common "look and feel" for the various screens or formats in the system? As the user moves from screen to screen, a common color scheme, layout, and general appearance minimizes user confusion.

Evaluating Design Parameters and Compatibility

A client, project manager, or other systems professional may wish to examine design parameter and compatibility issues. The demo used to examine these issues may move sequentially through all or the most critical individual functions with little or no script. This type of demonstration explores options and flexibility more than organizational fit and can address the following questions:

- What features are supported? The team may provide a full list of functions and demonstrate the most critical ones. A client demonstration can stress the features requested by the client during the problem definition stage.

- How are the functions implemented? What options are available and how are they accessed? A demonstration might take the most critical function and work through all of the available options.
- How does the system fit with the current IT environment? Clients and managers may wish to know how the proposed system will interact with existing systems, data, and infrastructure. A new system might extract data directly from an operational data store or may require special new hardware and/or the execution of a special-purpose extraction program for data access.

A POC demonstration also can provide confirmation to clients and sponsors that the system will produce the results they expect, particularly regarding issues of accuracy and validity. A common way to demonstrate accuracy in a POC is (1) either enter or display a small set of input data and (2) run the appropriate function in the system to verify that the output is as expected. Errors discovered in this type of presentation can raise serious questions about the credibility of the whole system. Questions for accuracy and validity include the following:

- Does the system replicate known results? Clients want to start with the answer and verify that the system gets the same result.
- Does the system behave correctly in the presence of known invalid data? The presentation can include examples with invalid blank fields, improper negative entries, impossible dates, and other problems to show that the system will not accept the invalid data and produce a misleading result.
- Are all results within intuitive bounds? Some things are difficult to estimate exactly, but many clients know the difference between reasonable and unreasonable answers. In the case of GB Video, a typical customer may rent one or several videos at a time, but a report that shows that a typical customer rents 73.2 videos at a time probably represents a system design or input error.

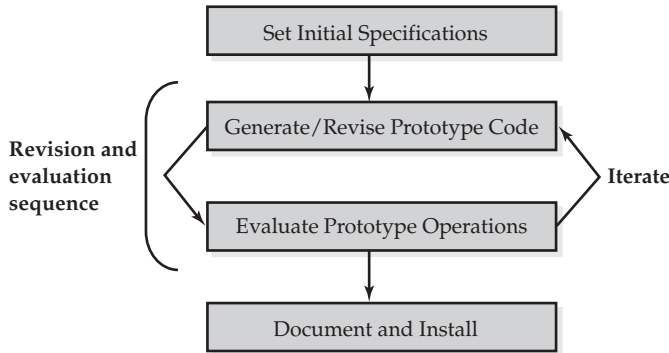
PROTOTYPE-BASED DESIGN

In addition to serving as a POC model, a prototype can serve as a supplement or alternative to a traditional SDLC project approach. A prototyping design methodology can reduce or replace much of the documentation and analysis of the SDLC approach. The **prototyping life cycle** shown in Figure 12.1 begins with an initial analysis to identify the data and process specifications—a brief informal proposed systems stage.

To devise the initial specifications, the team can follow in abbreviated fashion the following SDLC-type steps:

1. *Obtain a general understanding of the problem and task.* Normally, much of this understanding will come from conversations with the client. The client identifies the features the proposed system should possess. Reviewing relevant documents and observing the current operation may help.

FIGURE 12.1
The Prototyping Life Cycle



2. *Scan the environment for ideas.* Do team members have relevant experience? Do similar applications exist in the literature or at organizations that are accessible? If so, copy the good ideas from them.
3. *Use the information collected to set the initial specifications.*

The analysis provides a starting point only; the prototype iterations that follow will help the team to discover a more complete and correct set of design specifications. However, the more complete the set of features provided by the client, the better the team can select the initial specifications and the faster the development will proceed. The team builds the initial prototype to match the design specifications. Once the prototype operates correctly, the team uses the prototype to interact with clients, often including users. The team runs the prototype for the clients and encourages comments, corrections, and complaints. The team can forestall a number of problems by managing client expectations: The team should stress that the prototype is a tool for generating ideas and not a final product.

For the approach to work, the team needs to find ways to obtain feedback from the clients. Often the clients provide feedback without much prompting. If the team invites the clients to provide sample input for the prototype, the team should offer to key in the input for managers and sponsors. Some managers and sponsors may be offended or embarrassed by requests that they act as data entry clerks. Users may wish to key in the input themselves.

While the prototyping life cycle approach can contribute to any system development project, the way the team uses the prototype will depend on the task. In developing an order entry system for a catalog store, for example, the team may spend a great deal of time interacting with order clerks. For an e-commerce site, the team may need some simulated users probably nontechnical staff members acting as customers. For a utility billing system, the team may interact a little with meter readers or data entry clerks but interact more with relevant managers and project sponsors.

The team records the comments from each prototyping session and rapidly implements changes in the model to address them. Every several days to several weeks, the revision and evaluation sequence is repeated until the client is satisfied, usually after three to five iterations. Some prototyping methodologies

use one-day turnaround with small scope changes. The many-iteration methodology can generate dozens of iterations and can serve well for internal team use. However, large numbers of iterations probably will exhaust quickly the time and patience of the clients. In addition to interacting with clients, the team can use the prototype to examine such technical design issues as network and infrastructure ideas, database design, and response time. Some evolutionary prototypes provide a good vehicle for testing design parameters and benchmarking response times.

The initial specifications and prototype usually include a fairly complete and populated data model. While the team can prototype data model development, data prototyping generally takes more time than just preparing a “good guess” data model and can make initial client reviews confusing. The initial data model should not be viewed as a final version but should be complete enough to support the iteration exercises for the most essential functions of the system. The data model must support the generation of outputs that are realistic enough to allow the team and client to identify function, accuracy, and other problems and issues with the design. A one-line report for one customer making one error-free transaction probably will not allow the clients to determine if the system has the features they want. When feasible, the team can ask the client to provide the data to populate the data model. Actual data from operations or client-generated data often will exercise the prototype better than data the team generates.

An initial prototype also needs substantial functionality in place for clients to respond in a meaningful way. The prototype must execute at least the core functions in a manner realistic enough for the clients to identify the things they like and dislike. Although the initial prototype may implement only selected functions and options, the functions and options included should work correctly.

Once the team and client are satisfied that the current version of the prototype meets all the essential requirements, the prototype represents the final design specifications and may represent the production code for the system. The team may wish to add additional internal error checking and recovery features and include features to enhance multiuser operation, response times, and efficient operations. As a final step, the team generates the appropriate documentation including implementation, testing, and maintenance plans.

Prototyping can offer a useful alternative or complement to the SDLC approach for some projects. Many of the same steps occur with prototyping and traditional approaches, but the steps occur in different time sequences using different models and mechanisms. For smaller systems and for projects where the client cannot verbalize clear requirements for the system, the prototyping life cycle may offer the most effective alternative. Prototypes also can serve a useful role to complement other steps and mechanisms in SDLC type approaches. As discussed, prototypes can serve as POC models. Some projects include a prototyping phase to clarify user requirements and then follow traditional paths to define specifications for the proposed system and for system delivery. Boehm includes prototyping as a component used in every cycle in the spiral model discussed in Chapter 3.

BUILDING A PROTOTYPE

Building a prototype involves a development subproject within the overall project set forth by the client. Fortunately, the team may have completed some of the work. The team may have developed the conceptual specifications for the prototype following the guidelines in Chapter 8 and extended the work to logical and physical specifications using the guidelines in Chapter 11. Otherwise, the team needs to define a set of specifications that will meet the client's requirements. With initial specifications in hand, the team can proceed to select a focus, prepare a database, write code, and test the result. Team members with a lot of knowledge and organizational experience often can use a one-stop shopping method—develop the specifications and select the focus while sitting at the console writing the code.

Choosing a Focus

As discussed, prototypes come in different forms for different purposes. Before starting actual construction, the team chooses the focus for the design. As part of the focus the team chooses (1) either a throwaway or evolutionary plan; and (2) the component emphasis: output, process, input, data, or often a combination. The choice between an evolutionary or a throwaway prototype involves two basic issues: purpose and resources. Some clients want the team to provide only design specifications. Client choice, policy, and/or culture dictate that the IT staff, or a vendor, will build the production system. In these cases, the team clearly wants to focus on a throwaway prototype. Other clients either want or are willing for the team to build and deliver the production system which suggests the choice of an evolutionary prototype.

Evolutionary Model Issues

When client considerations suggest the choice of an evolutionary prototype, the team should make a best effort attempt to comply. With a small, simple system and familiar technologies, the team probably can produce an acceptable evolutionary prototype. With a large system and/or unfamiliar technologies, the team should conduct a careful analysis of feasibility before proceeding. For reasons including time, team skills, and availability technologies environment, the team may decide that an evolutionary prototype is infeasible and choose a throwaway prototype even though the client wants a production system. The team should explain and discuss the decision with the client and if possible obtain client agreement.

Time constraints may not allow for the construction of an evolutionary prototype. A student field project exercise is constrained by the end of a semester and a team in an organization may face similar constraints. Every project is a learning experience and competent team members should expect to stretch their technical skills if necessary. But when the stretch becomes too large, the team may never reach the point of producing an evolutionary prototype or may produce a poor one. Developing systems in such “industrial-strength” languages as C++ or Java to production code standards may present an infeasible task. Even

when the team can build the prototype, the need to change the prototype can pose a problem. The team may need to change the prototype system a number of times in response to unanticipated problems and client requests. The complexity of the evolutionary prototype may require more time and effort for changes than is available.

To produce a quality result within the relatively short time span available, the team members may need to work in a language and environment that one or more members know or can learn rapidly. For this reason the team may choose to use a language, for example, Visual Basic (VB), which facilitates rapid development and change, but may not produce efficient production code. A final issue concerns the availability of the technology environment. The team may not have access to the technologies the client wishes to use in the production system. The client may agree to provide access and technical support at the client's location. Unless the team is able and willing to spend substantial amounts of learning and building time at the client's site, the likelihood of completing a satisfactory evolutionary prototype is small.

Model Content Issues

The team also chooses a **content** emphasis for the prototype. Many prototypes emphasize the output portion of the system, although often not in the language that the production version will use. A team may develop a user interface model in MS Access and use a more powerful language such as DreamWeaver or Java for the production version. When the purpose of the prototype is a POC model to test the effectiveness of communication screens, the rest of the model may consist only of modules that feed responses to the screens. While the various prototype emphases are discussed separately below, the actual prototype may contain several emphases in different degrees.

A process prototype can demonstrate the logic of the proposed system. The critical issue in such a prototype is assuring that the internal logic in the design specifications for each module works as intended. The process prototype also can check module communication or message passing to assure that the control and data flows are correct. The team should note any changes immediately in the design specification documentation. The prototype will normally focus on the basic function logic and add edit, error checking, and logging functions later to the prototype or directly to the production system.

An input prototype models the data capture and interface functions of the system. The client's usually does not wish for the team to interface the prototype with other in-use production systems until the prototype reaches a fully tested production status. A data capture prototype usually tries to interface with a test program and data set that contain the relevant features of the real data stream. The client's IT staff may provide the test programs and data set. The test data set may consist of a small portion of data from the actual data stream. The prototype can confirm such design issues as formats, input logic, handling of unusual data inputs, and data updates. Once the team is satisfied that the basic input structure works correctly, the team can add functions that manage flow validation, error detection, logging for normal processing, and exception reporting.

A data prototype can help the team to develop and exercise the data model for the system. Such a model normally creates the initial data structure for the system with a limited set of attributes. As the team uses the model with the client and learns more about the client's requirements, the team can identify and add additional attributes, sizes, and formats, and confirm constraints. When the prototype includes processing components, the team can assist with identifying appropriate views and explore denormalization. The learning from the prototype goes into the design specification documentation. When the documentation exists online inside the database engine, the documentation probably updates automatically when the team makes changes or additions.

Making Initial Design Decisions

To the extent feasible, the team wants to make design decisions for the prototype that will allow for "rapid development." As noted earlier, the basic tenant of rapid development is, "Do only what is necessary to deliver an application that (1) meets the client's perceived needs; (2) in as short a time as possible; and (3) at the lowest possible cost." The following rules and guidelines should facilitate rapid development. The team must look carefully at each rule to decide if the rule is feasible given the team and client goals for the prototype. Using Access can offer a simple and fast approach but is infeasible if the team agreed with the client to use Java and Oracle.

1. *Select a technology environment—infrastructure, language, and tool set—that enables the team to rapidly build and modify the prototype.* Microsoft Access provides input and output forms and allows easy, direct entry of test data into tables. Visual Basic can simplify the creation of full-function input screens. Tools for Web site prototypes range from simple HTML to MS Front Page type applications to packages that provide the framework for a catalog and retail order fulfillment system. When Web pages read from or write to a database, Active Server Page (ASP) or similar tools can help. A throwaway prototype development environment may lack the efficiency, capacity, and security required for a production system.
2. *Focus on the key functionality.* Include the features that are of primary or most interest to the client; leave out other features, at least, at the beginning. Key features may include processes to enter external input, transform it, store and retrieve data, and produce reports.
3. *Include only limited edit and error handling capability.* In a throwaway prototype, these functions provide little value added unless they are the focus of client requirements. In an evolutionary prototype, comprehensive error checks make more sense after the team refines the basic functionality.
4. *Put extra effort into input and output formats.* Generally, the input screens and the report or output screens are the things of most interest to the client.
5. *Include a realistic and largely complete data schema.* Remember that data tends to change more slowly than processes. For example, relational tables often will transfer with little change to the final system. The prototype schema should include all of the major tables in the final system. Normally, the schema includes all the attributes of the final schema; but in complex situations, the

team may omit some attributes in the initial version. Use mandatory (minimum cardinality of one) relationships sparingly in data tables. Mandatory relationships can make populating the tables with data more difficult.

6. *If possible, populate tables with adequate but small data sets.* An actual personnel system may hold records for several thousand employees, but the prototype may serve a POC role well with records for 10 employees. The prototype for a POC model of a retail catalog system may need 10 products, three sizes, and four colors, not the thousands of choices found in a real catalog.
7. *Approximate or stub complex logic.* The actual logic to evaluate a mortgage application might involve tens of variables and maybe fuzzy logic or neural nets. For a first-pass prototype, a simple scheme that rejects the bad and accepts the good may be sufficient. Use a corresponding data set with only very good and very bad applicants.
8. *Omit, initially at least, features that add integrity to the production system—high efficiency transaction processing modules, exception or operational logging, security, and so on.*

Generating Code

Generating code may involve such dimensions as module development, integration of modules, schedules, and work assignments. Module development focuses on (1) identifying and coding the modules and (2) work assignments. Each of the boxes on the process hierarchy chart (see Chapter 8) provide one approach to identifying modules. Many times, the team will assign several people to code different modules or even a single complex module. The team needs some process to assign modules to members and to develop a time and sequence schedule for module coding.

Schedules and Assignments

Most projects have delivery deadlines that require finding ways to assign module coding to several people at once. Assignment and schedule strategies include the following:

- A **top-down plan**. A system that follows structured design principles uses a structure chart with a master control module at the top. The top-down plan builds the master control module first and develops the control and data transfer protocols to communicate with the next level of modules. With the control structure in place, different people can develop the next level modules. Coding proceeds down the diagram until finished. A similar strategy works for a dialog-driven system with a hierarchical navigation structure.
- A **bottom-up plan**. With a bottom-up plan, the team assigns related groups of the lowest level or elementary modules to the members for coding. When the elementary groups are complete, the team assigns people to code the next level until the work reaches the highest level. This approach depends on complete and accurate design specifications to avoid writing and rewriting the lower level modules as the team progresses and thus works best in a traditional SDLC project.

- A **component** or **object plan**. An OOD-type design documents the complete set of objects or components and the message interfaces for the objects. With an object design, the team can assign objects to different team members. Each team member then codes the object or, when possible, modifies an existing generalized object to meet the design specifications. Because modules are encapsulated, OOD provides a good framework for dividing up the work.

Coding and Design Specifications

During the proposed system phase discussed in Chapter 8 and the system design activities described in Chapter 11, the team prepared design specifications, extended the specifications into a data schema and a set of process modules, and prepared metadata for the schema and modules. A team that plans to use the prototyping life cycle will have to prepare the initial design specifications before proceeding.

Modules of code are written to execute all the functionality of the module and may range from a few lines to several pages of code. Some early textbooks suggested that a module should be equal to or less than 24 lines. With the advent of nonprocedural languages and object-oriented code, this restriction makes little sense. The team should test each module as it is written, test module integration, and test the complete prototype following the guidelines listed in Chapter 13. Clients should develop or participate in developing the test data.

One of the primary purposes of a proof of concept model is to demonstrate that the design decisions made by the team actually work. The two matching rules are as follows:

1. *The initial prototype code that the team produces must match the design documentation.* The design documentation is the team's formal specification of how to solve the client's problem. If the design does not match the code, the team and client may spend a large amount of unnecessary time and effort determining whether any problems that occur reflect the original specification design decision or a new coding design decision.
2. *Any changes the team makes to the prototype code should be made to the documentation.* The design documentation may become the maintenance guide for the system. As a maintenance guide, the documentation must correctly reflect the "as built" system and allow the maintenance programmer to understand the code architecture and locate the spots in code that execute specific functions.

A POC model of any sort should accurately reflect the design documentation. In particular, the code and the design should have:

- Identical control flow.
- Identical data model.
- Like naming.
- Corresponding data access.
- The same execution logic.

Control Flow

The first requirement of a maintenance programmer is to understand the control flow of the program. **Control flow** refers to the sequence in which the different program components are executed in response to program input. The actual prototype components are **artifacts**—pieces of code that perform a program function. A good proof of concept model makes evident the correspondence between the structure in the design and the structure in the code. The model artifacts should match lines of pseudocode of functions described in the design.

Artifacts differ from language to language. In such procedural languages as FORTRAN and COBOL, the artifacts correspond to subroutines and modules on the program structure chart. In such environments as Visual Basic or MS Access, the building blocks are screens or forms and controls on the forms. Page navigation maps can describe design specifications for these environments. Regardless of the environment, the triggers and sequence of execution for each of the artifacts should match the design.

In most cases, the code artifacts—subroutines, functions, forms, or controls—have names. These artifact names should match the names for the corresponding functions in the design documentation. Choosing names with appropriate prefixes can help readability by generating a sorted listing of code artifacts that presents them in a logical order. In a MS Access implementation of the GB Video system, the module “2.0. Create or update member data,” in the program structure chart for GB Video can correspond to a form named “20CreateOrUpdateMemberData.” Most programmers suppress blanks in variable names in code to simplify code writing. The prefix 20, 21, and so on the variable name for artifacts corresponding to modules, 2.1, 2.2, and so on will cause an alphabetic sort of form names to appear in an ordered list. Microsoft coding convention adds the prefix “frm” to the names of all forms. Thus, the name used in the program code is “frm20CreateOrUpdateMemberData.”

Data Model

A prototype model may use a **data model** that differs from the intended operational database for the proposed system for a number of reasons. The real model may be too complex or may be part of a much larger system to which the team does not have access. Developing test data for the actual system may consume too much time. In these cases, the team produces a new data model for the POC and explains the differences and reasons briefly in the documentation.

The original design data model, the revised model, if any, and the prototype implementation should match as closely as possible. For the tables and attributes that appear in the prototype, all the representations should contain consistent table names, foreign key relationships, referential integrity rules, and attribute names and types. The actual system may record more metadata than the design model, but the metadata present in both should match.

Naming

In addition to data and control structures, all the other names specified in process metadata should correspond to the names used in the actual code, that is, the

core name should match in all representation but prefixes may vary. The pseudocode example in Chapter 11 uses a prefix letter “I” to indicate variables input from an external source. The actual code may use other prefixes or conventions to show input variables. Microsoft coding practice adds such control prefixes as “frm,” “cmd,” “txt,” and others to indicate the type of control on the form that implements the variable.

Data CRUD Operations

The data operations logic in the prototype should follow the logic in the design documentation. In particular, table **create, update, retrieval, and delete (CURD)** actions should match the design documentation. The SQL-like statements in the actual code should resemble but probably will not match exactly the pseudocode. Pseudocode need not follow all of the correct syntax and can use English phrases in place of complicated SQL statements. The names of views or queries in MS Access should match the documentation.

Logic

One of the primary functions of good design documentation is to specify the processing **logic**. The code in the POC model should follow the design documentation logic as closely as feasible. While considerable differences may exist between the pseudocode and the actual code statements, the resulting functionality should match.

Effective Coding

The keys to successful programming are as follows: (1) make sure the code does what the user wants; (2) use the built-in tools in the language whenever appropriate; (3) use code that has already been tested when available; and (4) test, test, and test some more using the procedures given in Chapter 13. Much coding consists of a throughput effort where the code is developed, used, patched, or thrown away as the system modules are integrated. Reusable code can save valuable programmer time. Many programming tools contain higher-level instructions or macros for a number of common functions that may appear in modules. Some organizations maintain libraries of reusable code. The team should use code from the libraries when feasible, but code modules may not be available in the language chosen for the prototype.

A GB VIDEO PROTOTYPE

Chapter 11 contains tables and figures with the detailed data and process specifications required to build the production version of the proposed system. These specifications also can serve as the specifications for a proof of concept prototype. This section of the chapter illustrates how a team can convert the specifications into a prototype. The GB Video prototype uses Microsoft Access as the physical tool and represents a throwaway prototype. The team can use the prototype to demonstrate system features to the client and to refine the specifications. Mr. Cosier and GB Video will either purchase a package or contract with a vendor to obtain the production system.

Building a prototype in Access or a similar tool normally consists of six steps:

1. Create the tables defined in the relational schema.
2. Populate the tables with a small set of sample data.
3. Set up a switchboard.
4. Design the input forms.
5. Design the reports or output forms.
6. Insert logic into the forms as needed.

Each of the steps applied to the GB Video prototype is described in the following materials. The materials provide only an overview on creating and populating tables and creating forms, modules, and code. Access, while relatively easy to use, is a physical tool with very specific features and conventions. Consult the Help screens and/or an Access book for detailed information and a full set of options.

Creating the Tables

Chapter 11 contains examples with the relational schema and metadata for the GB Video database. The database consists of four tables: CUSTOMERS, RENTALS, LINES, and COMVIDEOS. The relational schema and metadata used for the prototype appear in Figure 12.2.

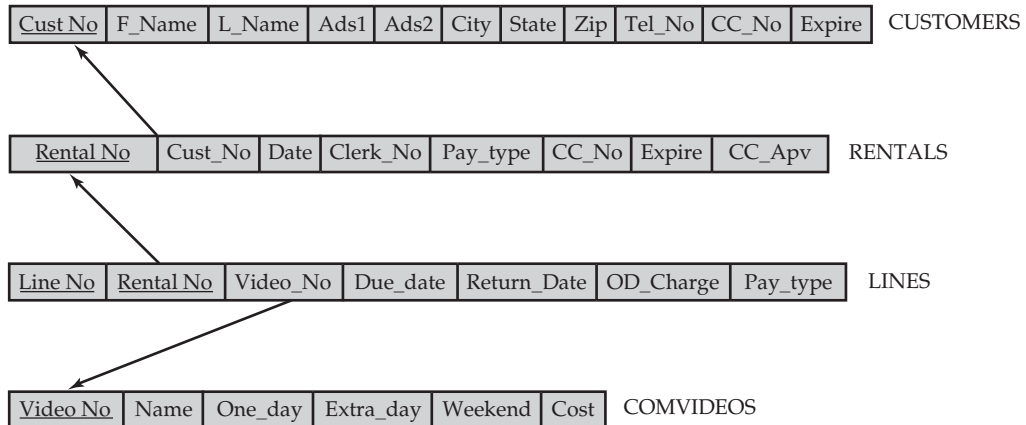
Several ways exist to create a new table in Access. This example describes creating tables from the Design view. In the Database window, select “Tables” on the Objects list and either click “New” on the toolbar or double-click “Create table in Design view.” The table design view window should now read “Table: Table1” and show columns “Field Name,” “Data Type,” and “Description.”

Creating the Table Design

Start the table design by entering the field names, descriptions, and the data types shown in the metadata. For example, start with the GB Video Customers table. The metadata contain a description for each of the attributes in each table. The selection of data types depends on the application and the usage of the attribute. For example, to store a name, Social Security number, or a combination of text and numbers, use the “Text” data type. Characteristics of data types include

- Field size. The number of allowable characters in data type.
- Input mask. A specific formatted data for the field (for example, (405) 555-4565 for phone numbers, date formats, etc.).
- Caption. Label for the attribute in forms.
- Default value. An automatic, standard value field.
- Validation rule and validation text. The limit enforced on the data range the user can enter into a field and a text error message when the user violates the limit.
- Required. An indicator to show if the attribute can be null.
- Indexed. An indicator to show if the attribute is indexed or not, and whether duplicates are allowed or not. The primary key for the table is indexed with no duplicates at default.

FIGURE 12.2 Relational Schema and Table Metadata



Metadata for CUSTOMERS

Data-Item	Description	Optional	Type	Size	Decimal
Cust_No	A unique identifier assigned to each customer (PK)	No	Text	10	
F_Name	First name and middle initial if any	Yes	Text	15	
L_Name	Last name	No	Text	30	
Ads1	Street or box address	No	Text	30	
Ads2	Apartment number or other as needed	Yes	Text	30	
City	Name of city	No	Text	20	
State	State id code	No	Text	2	
Zip	Zip code	No	Text	9	
Tel_No	Telephone number	Yes	Text	10	
CC_No	Credit card number	No	Text	16	
Expire	Expiration date on the credit card used to enroll	No	Date		

Metadata for RENTALS

Data-Item	Description	Optional	Type	Size	Decimal
Rental_No	Unique identifier assigned to each rental (PK)	No	Auto-number		
Cust_No	The customer for the rental (FK)	No	Text	10	
Date	Date of the rental	No	Date		
Clerk_No	Employee number of the clerk entering the rental	No	Text	3	
Pay_type	Cash, check, or credit card	No	Text	1	
CC_No	Credit card number	Yes	Text	16	
Expire	Expiration date of the credit card used for the rental	Yes	Date		
CC_Apv	Credit card approval code	Yes	Text	6	

Metadata for LINES

Data-Item	Description	Optional	Type	Size	Decimal
Line_No	Unique identifier assigned to each line (PK)	No	Auto-number		
Rental_No	The rental number that this line belongs to (FK)	No	Long Integer	10	
Video_No	Video rented on this line (FK)	No	Text	14	
Due_date	Date video is to be returned	No	Date		
Return_Date	Actual return date	Yes	Date		
OD_Charge	Charge for days kept after due date if applies	Yes	Currency		2
Pay_type	Method of payment for the overdue charge	Yes	Text	1	

Metadata for COMVIDEOS

Data-Item	Description	Optional	Type	Size	Decimal
Video_No	A unique identifier assigned to each video (PK)	No	Text	14	
Name	The title of the video	No	Text	30	
One_day	First day rental fee	No	Currency		2
Extra_day	Extra days rental fee	Yes	Currency		2
Weekend	Rental fee for Sat. and Sun.	Yes	Currency		2
Cost	Price GB paid for the video	No	Currency		2

Page 3

Table 12.1 shows some common data types and characteristics.

The relational schema and metadata identify the primary key (and/or foreign keys) for each table. To create a key for the Customers table, click on the far-left gray box of the Cust_No line. Then click the Key icon in the toolbar at the top of the screen. For a composite key, hold the Control key down and click on all the necessary key attributes to highlight the area, then click on the Key icon. With the AutoNumber data type, Access automatically generates a unique index on the primary key field to guarantee the uniqueness of the key values. If the user does not specify a primary key, Access prompts for permission to create a key for the table. If the user selects “Yes” when prompted, Access generates an ID column as the primary key. When the user selects “No,” Access creates the table without a primary key. The data type of a primary key must match the data type of the foreign key, except an auto number primary key matches with a number-type foreign key.

To save the table design, follow normal Microsoft conventions: either click on the Save icon or click “File” and “Save” on the toolbar.

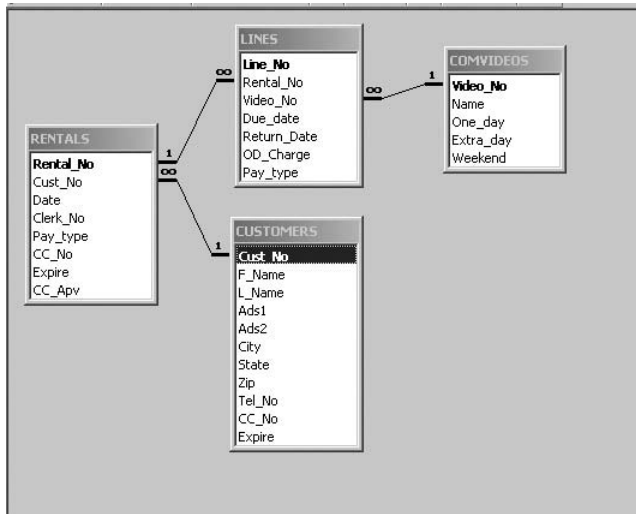
Creating Table Relationships

To begin, either select “Relationships” from Tools on the menu bar, or return back to the database window and click on the Relationships icon. In the Show Tables

TABLE 12.1
Data Types
and Characteristics

Data Types	Characteristics
Text	Appropriate for data containing text, combinations of text and numbers, or numbers that are not used for calculations.
Number	Appropriate for data used for calculations.
AutoNumber	Most appropriate for primary keys.
Date/Time	Appropriate for date and time data and calculations.
Currency	Appropriate for currency values, and is most accurate in calculations.
Yes/No	Appropriate for Boolean—true/false data types

FIGURE 12.3
Table
Structure for
the GB Video
Database



window, select the tables by double-clicking the table name. If the relationship involves a query, click on the Query tab and double-click on the query name. The relationship view now shows all the selected tables.

To connect the tables, click and hold on the primary key from the “one” table, and drag to the corresponding foreign key in “many” table. For example, drag the Cust_No attribute from the Customers table to the Cust_No attribute in the Rentals table. Access creates a relationship line between the two attributes. In the Edit Relationship window, check the “Enforce Referential Integrity” check box. The window also shows the relationship type—one-to-many, one-to-one, and others. Other options include the Join Properties. To select Join Properties, click on “Join Type” and select the joins desired. Figure 12.3 shows a screen shot from Access of the tables and table relationships in the GB Video prototype.

Populating the Tables

Once the table structure exists, the team can add data to or populate the tables. Access allows team members to type input data directly into the tables. Figure 12.4 shows a screen shot from Access for the Comvideos table populated with test data to evaluate code execution. The test table omitted the attribute Cost, which appears in the data schema but plays no role in the program testing. The team can add the missing attribute and populate the tables with live data or with a larger generated data set later as a part of the test plan.

Coding the GB Prototype

The GB Video structure chart in Chapter 11 shows the following modules:

- 1.0 GB Video Rental and Return Switchboard
- 2.0 Check and update data or enroll a new member

FIGURE 12.4
Populated
Comvideos
Table

COMVIDEOS : Table						
	Video_No	Name	One_day	Extra_day	Weekend	
	+ 1	Titanic	\$1.00	\$2.00	\$1.50	
	+ 2	Total Recall	\$1.00	\$2.00	\$1.50	
	+ 3	Independence Day	\$1.00	\$2.00	\$1.50	
	+ 4	Bull Durham	\$0.75	\$1.00	\$0.90	
	+ 5	Gone With the Wind	\$0.75	\$1.00	\$0.90	
	+ 6	Ray	\$1.50	\$2.00	\$1.50	
	+ 7	Rocky Horror Picture Sho	\$0.50	\$0.50	\$0.50	
	*		\$0.00	\$0.00	\$0.00	

- 2.1 Input the data for a rental
- 2.2 Calculate cost and process payment
- 2.3 Create rental and lines
- 2.4 Print a receipt
- 3.0 Record return of video(s)
- 4.0 Identify overdue videos

The Access prototype implements these modules as forms or modules with names that resemble the module names in the structure chart. Figure 12.5 shows a list of forms (frm10, frm20, frm21, frm22, and frm24) that correspond to the structure chart modules 1.0, 2.0, 2.1, 2.2, and 2.4.

Figure 12.6 shows the Access modules. Structure chart module 2.3 becomes an Access module named *Module23CreateRentalAndLines*. Access *ModulePostCCTransaction* is part of the pseudocode in the metadata for module 2.2 on the structure chart

FIGURE 12.5
Access Forms
for the GB
Video
Prototype

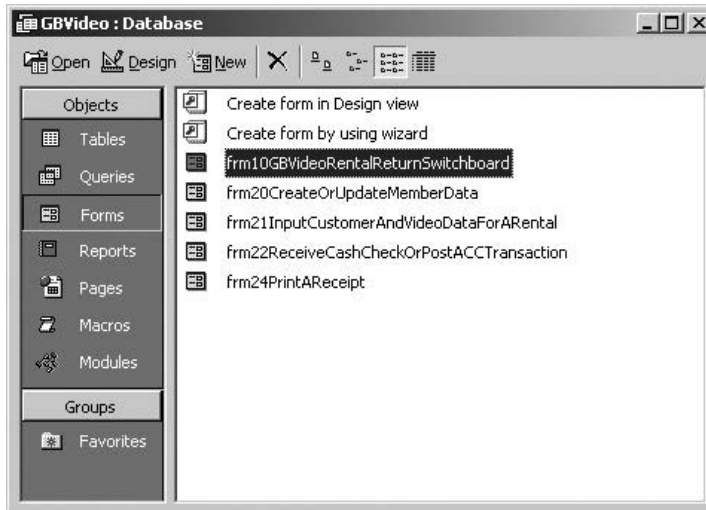
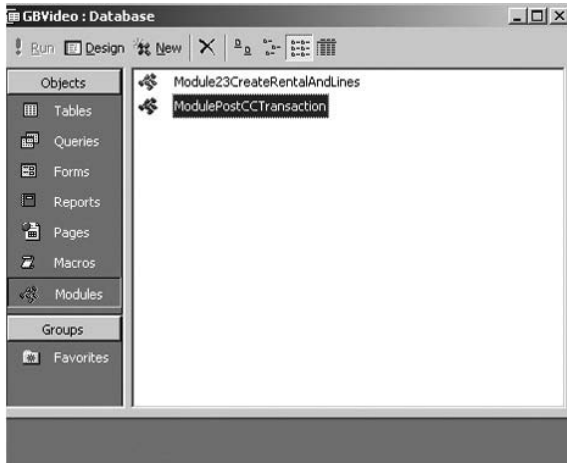


FIGURE 12.6
Access
Modules for
the GB Video
Prototype



and represents a call to a function supported by the credit card company. Modules 3.0 and 4.0 are not included in the prototype. The diagrams after Figure 12.6 show the Access code for the forms and modules.

The Switchboard

Figure 12.7 shows the GB Video system switchboard implemented by *frm10GBVideoRentalReturnSwitchboard*. The switchboard is simple to create and uses three buttons to select each of the subsequent modules. Once the buttons are in place, the team writes the code for each button to open the correct module.

Clicking the button *Create or Update Member Info* triggers or opens the form called *frm20CreateOrUpdateMemberData*. As part of the trigger, a message box opens to obtain input regarding the customer status: old or new. Two private

FIGURE 12.7
The GB Video
System
Switchboard

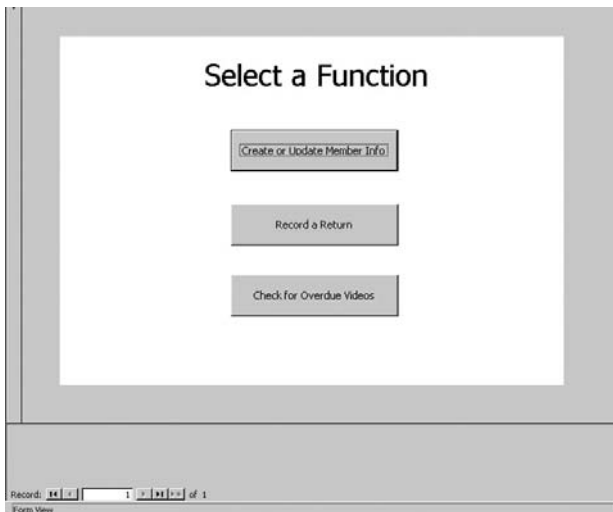


FIGURE 12.8
Old Customer
Screen

The screenshot shows a form titled "GB Customer Record" with the following fields and labels:

- First Name (text box)
- Last Name (text box)
- Member Number (text box)
- Address 1 (text box)
- Address 2 (text box)
- City (text box)
- State (text box)
- Zip (text box)
- Credit Card Number (text box)
- Phone (text box)
- Expiration Date (text box)

At the bottom of the form are two buttons: "Look up Member" and "Update Member Data and Proceed to Checkout". The status bar at the bottom indicates "Record: 1 of 1" and "Form View".

subroutines, *PerformOld* and *PerformNew*, open the member data form for either a returning customer with data or a new customer without data. The message box implements the “Input (I_Customer_status) and Select (I_Customer_status)” pseudocode for module 2.0 on the structure chart, an example of how the programmer needs to show creativity to implement a system. The box represents the physical or Access version of the logic in the design documentation.

frm20CreateOrUpdateMemberData

Figures 12.8 and 12.9 show the data entry screens that correspond to module 2.0 in the design documentation. Figure 12.8 shows the screen for an old or a returning customer. The form fields correspond to the database attributes or columns in the Customers table. In addition to the data fields, the form needs two buttons.

FIGURE 12.9
New
Customer
Screen

The screenshot shows a form titled "GB Customer Record" with the following fields and labels:

- First Name (text box)
- Last Name (text box)
- Member Number (text box)
- Address 1 (text box)
- Address 2 (text box)
- City (text box)
- State (text box)
- Zip (text box)
- Credit Card Number (text box)
- Phone (text box)
- Expiration Date (text box)

At the bottom of the form are two buttons: "Look up Member" and "Create New Member Entry and Proceed to Checkout". The status bar at the bottom indicates "Record: 1 of 1" and "Form View".

The form and the code attached to the buttons implement the pseudocode functions for module 2.0.

1. *Look up Member* searches for an existing customer based on customer or telephone number. The button creates a query based on the customer or telephone number and displays the matching customer information. The prototype allows the clerk to search through the data for all (if more than one) of the customers who match the search criteria until the desired one is found. Once the correct customer data appears, the clerk can modify any of the data, except the customer number, by typing new entries in the data boxes.
2. *Update Member Data and Proceed to Checkout* commits changes to the customer data, if any, and then continues to the checkout. The code updates the CUSTOMERS table in the database with changes made in the form and then triggers or calls the rental checkout form, *frm21InputCustomerAndVideoDataForARental*, passing the customer name and number data to the new form. The physical design implements Mr. Cosier's requirement: the only entry into rental is from the member form with a valid member number.

Figure 12.9 illustrates the display for a new customer. The form resembles the old customer form except the Look up button is not active and the commit button changes to *Create New Member Entry and Proceed to Checkout*. The code for the Create button is exactly the same as described for the old customer screen. The production system probably will use one customer screen for both old and new customers, and the prototype could have used one screen with a slight design change. A major purpose of the prototype is to learn about design changes to improve the system.

frm21InputCustomerAndVideoDataForARental

Figure 12.10 shows the video rental form that corresponds to module 2.1 on the structure chart. After opening, generating a rental number, and displaying the

FIGURE 12.10
GB Video
Rental Form

FIGURE 12.11
GB Video
Checkout
Form

customer initialization data from the customer form, the rental form accepts input from the clerk for the employee number. In the prototype, the *Add Video to Checkout* button opens a message box in which the clerk manually enters each video number. In the production system, the video numbers will come from a scanner. A list box in the center of the form displays the videos selected by the customer. A box in the lower right corner of the screen keeps a running total of tax and total price for the rental. In the prototype, a 100-item single-dimension array holds the video numbers for each rental, probably providing more space than is needed. The *Process Payment and Checkout* button triggers the checkout form and passes all of the information collected thus far to the checkout form.

frm22ReceiveCashCheckOrPostACCTransaction

Figure 12.11 shows the checkout form which executes the functions in structure chart modules 2.2, 2.3, and 2.4. This form has three buttons to indicate (1) a cash payment, (2) a check, and (3) a credit card payment. The Cash and Check buttons trigger the create rental and lines module and pass the payment type data plus the other data. The Credit Card button (1) calls a stub module that in the future will process the credit card transaction and return an approval code; and (2) triggers the create rental and lines module. The create rental and lines module, explained in the next section, commits all of the data in the form, that is, it stores all of the rental transaction data in the database (structure chart module 2.3). After data is committed to the database, the code in the form opens the receipt form, explained after the create module, and passes the rental number to it, and calls its print subroutine (structure chart module 2.4). The clerk and customer never see the “hidden” print form; the customer gets the result—the printed receipt. Finally, the buttons open the switchboard to return the user to the starting state.

Module23CreateRentalAndLines

Figure 12.12 gives the code for *Module23CreateRentalAndLines*, which corresponds to structure chart module 2.3. This module creates the new rental record in the

FIGURE 12.12
Access Code
for Module23-
*CreateRental-
AndLines*

```

General) SaveRentalInfo
Option Compare Database
' This function takes in the Rental table, the list
' of videos rented, and the number of videos rented
' and completes the creation of the rental record
Public Function SaveRentalInfo(rsRental As Recordset, listVideos() As String, intNumVideos As Integer)

    ' Update the Rentals passed in
    rsRental.Update
    rsRental.MoveLast

    ' Now that the Rental line has been created, create the rental lines
    ' table to enter the video data
    Dim rsRentalLines As DAO.Recordset
    Set rsRentalLines = CurrentDb.OpenRecordset("LINES")

    ' Now loop through the videos list and add them to the rental lines table
    For intIndex = 1 To intNumVideos
        ' Add a new line
        rsRentalLines.AddNew

        ' Now enter the rental number, video number and due date for the video
        rsRentalLines![Rental_No] = rsRental!Rental_No
        rsRentalLines![Video_No] = listVideos(intIndex)
        rsRentalLines![Due_Date] = Date + 3

        ' Now update the table to save the info to the database
        rsRentalLines.Update
    Next ' Loop to the next entry in the video list

    ' Clean up the recordsets
    rsRental.Close
    rsRentalLines.Close
    Set rsRental = Nothing
    Set rsRentalLines = Nothing
End Function ' End SaveRentalInfo

```

Rentals table and then iterates through the array of video numbers creating the line records for the rental in the Lines table.

frm24PrintARceipt

Figure 12.13 provides the screen shot of the receipt form, which looks similar to the rental form and corresponds to structure chart module 2.4. The receipt form code uses a query based on the rental number that was passed to it by the checkout form

FIGURE 12.13
GB Video
Receipt Form

GB Video Stores

Date: Emp #: Rental No.

Member: Member No.

Tax

Total

Record: 14 of 1

to (1) retrieve and fill in the rental information from the Rentals table and the Customers table; and (2) retrieve and fill the list box with the video information for the rental from the Lines table. Structure chart module 2.4 receives all of the needed data from module 2.3 instead of retrieving it. The prototype is much more parsimonious in the data it passes from module to module, a design that simplifies keeping track of what is happening. The print subroutine prints all the data in the form and then closes the form.

Summary

A proof of concept (POC) model provides a representation of the final production software for a project that allows the team and client to directly observe some characteristics of the system. An operational proof of concept model converts inputs into outputs, stores and retrieves data, and exercises some or all of the processes in the system. A static POC model illustrates the outputs that result from a predefined set of sample inputs. A proof of concept model may serve one or more of the following purposes:

- To verify operational feasibility.
- To improve or refine the system specifications.
- To evaluate compatibility with a physical infrastructure.
- To confirm system design parameters.

A POC model can demonstrate the two basic components of operational feasibility: functionality and usability. To test functionality, the analyst or client can present typical external inputs and observe the POC model to determine whether all of the desired features are included and that they work as intended. Many POC models also demonstrate the usability of a system—how easy or satisfying or “likable” the system appears to users.

Proof of concept models consist of two major categories: prototypes and packages. The team builds a prototype POC model; that is, the team sets up and populates a data schema and writes the program code. When the team recommends that the client purchase a package, either the actual package or a demonstration version of it may serve as the POC model. A package may require substantial work by the team before it will operate correctly or at all.

When the team builds a solution or when no demonstration version of a package system is available, the team may choose to build a prototype. The prototype may range from a full-featured initial version of the actual production system to a highly simplified version that only provides some input to output transformations. The team can create and use a throwaway prototype to develop, refine, and demonstrate the system. The prototype provides a tool to clarify the design and operations of the proposed system before the client invests in an expensive and less flexible production model. An evolutionary prototype evolves into the final production programs—the programs that are implemented in the organization to carry out activities for the client. In a number of field projects, especially in smaller, low-budget operations, the client may wish to use the prototype produced by the team for a POC model as the production system.

Once the team possesses a POC model, either a package or a prototype, the team can use the model to validate and refine the design. A prototype also can serve as a supplement or alternative to a traditional SDLC project approach. The prototyping life cycle approach begins with an initial analysis to identify the data and process specifications—a brief informal proposed systems stage. The team builds the initial prototype to match the design specifications. Once the prototype operates correctly, the team uses the prototype to interact with clients, often including users. The team records the comments from each prototyping session and rapidly implements changes in the model to address them. Every several days to several weeks, the revision and evaluation sequence is repeated until the client is satisfied, usually after three to five iterations.

To the extent feasible, the team wants to make design decisions for the prototype that will allow for “rapid development.” As noted previously, the basic tenant of rapid development is, “Do only what is necessary to deliver an application that (1) meets the client’s perceived needs, (2) in as short a time as possible, and (3) at the lowest possible cost.” One of the primary purposes of a prototype is to demonstrate that the design decisions made by the team actually work. Two design rules are:

1. *The initial prototype code that the team produces must match the design documentation.*
2. *Any changes the team makes to the prototype code should be made to the documentation.*

The GB Video prototype implemented in Microsoft Access closely matches the design specifications in Chapter 11. The Access tables correspond to the relational schema in Chapter 11. The forms and modules in Access correspond directly to the modules on the program structure chart in Chapter 11. Access conventions and the physical model result in some changes in naming and also changes in the way that modules execute.

Key Terms

artifact, 439	logic, 440	prototyping life cycle, 431
bottom-up plan, 437	object plan, 438	simulated POC model, 426
component plan, 438	operational feasibility, 424	static model, 424
content, 435	operational model, 424	throwaway prototype, 428
control flow, 439	package POC model, 426	top-down plan, 437
CURD operations, 440	production program, 428	usability, 424
data model, 439	proof of concept (POC)	
evolutionary prototype, 428	model, 424	
functionality, 424	prototype, 425	

Review Questions

Answer the following questions regarding these topics.

1. Proof of concept.
 - a. What is the purpose of a proof of concept model?
 - b. What is the difference between an operational model and a static one?
 - c. What are the two components of operational feasibility?

2. Types of models.
 - a. Describe three types of proof of concept models.
 - b. What can each type of model test? Not test?
3. Purchased systems.
 - a. What are the normal proof of concepts that are available for a purchased system?
 - b. What are the considerations in determining whether to use a version of a vendor demonstration system or rely on a simulated model?
4. Prototypes.
 - a. What is the difference between a throwaway prototype and an evolutionary one?
 - b. What are the advantages and disadvantages to throwaway and evolutionary prototypes?
 - c. Why would you ever build a throwaway prototype?
5. Features.
 - a. What are some key features that should probably be included in the prototype?
 - b. What features are normally not implemented in a (throwaway) proof of concept model?
6. Walk-through.
 - a. What is the purpose of a walk-through?
 - b. Who should attend?
7. Proof of concept demonstration.
 - a. Why should a proof of concept demonstration for a client follow a prescribed script?
 - b. How would you demonstrate system functionality, usability, features, and validity?
 - c. What should the team do if you are unable to obtain an operational demo?
8. Prototyping life cycle.
 - a. Describe the prototyping life cycle.
 - b. Discuss the use of prototyping within the context of the spiral development methodology.
9. Documentation.
 - a. What should match between documentation and the proof of concept model?
 - b. What should not match?
 - c. What are concrete checks that a team should go over to be sure that code and documentation match?

Critical Thinking Exercises

Individual Exercises

1. You are building a throwaway prototype for your client. The client needs a system to keep track of inventory and orders for a book warehouse. What tables would you include, and what fields would you include in the tables?
2. Create the tables in Individual Exercise 1 with metadata in Access or a similar physical implementation.

Group Exercises

1. Use Access or a similar tool to build a simple prototype for Individual Exercise 1 that will provide a report on the inventory available for any book by title or ID number.
2. Find a demonstration of a package on the Web and develop the data schema and module logic for it.

Chapter Thirteen

Project Completion

Chapter outline

Introduction

Testing Plans

Desk Checks

Walk-Through Tests

Design Specifications Walk-Through Tests

Operational Testing

Post-Implementation Tests

Documentation Clearance

Implementation

The Implementation Plan

Implementation Strategies

Direct Implementation

Parallel Implementation

Sequential Approaches

Training

Initial Training

Follow-on Training and Support

The Training Plan

Maintenance Plan

Documentation

System Controls

Disaster Plans

Post-Implementation Audit Plan

GB Video Implementation Plan

Closing the Project

Summary

Key Terms

Review Questions

Critical Thinking Exercises

Individual Exercises

Group Exercises

Reference

INTRODUCTION

Completing a project means different things for each manager and client. Some clients want only the specifications to purchase or build a solution. Others want an operational solution, that is, a system that meets their requirements, with installed and tested production code, and with good, complete documentation. In all projects, the team should prepare testing and implementation plans and provide adequate documentation for all aspects of the project. In some projects the team will carry out part or all of the testing and implementation activities. In other projects, IT or contractor staff members will perform or assist the team in this function.

One of the key problems in testing, conversion, and implementation involves the time frame. Preparation and/or execution of test and implementation plans fall in the last stage of the SDLC, near the end of the project. By this time the team probably is tired and may expend little time and effort on implementation issues. To address the problem, planning for testing and implementation should begin at the start of the project. During project definition, the analyst should ask such questions as (1) What problems or issues in the current situation should be addressed during testing? and (2) What problems or issues in the current situation might be alleviated or eliminated by training? During the proposed system phase, a number of additional issues related to testing and implementation will emerge. For example, the team may place major weight on finding a low-risk alternative, one that the client can implement with a high probability of success. Finally during system design or procurement, the analyst may take a number of steps to simplify the implementation and maintenance of the system.

Testing may produce valuable input for implementation. Any problems detected during testing that remain in the production system should be addressed by the team in the implementation and training plan. For example, if the client decided not to include error checking on certain data inputs, the design team can point out how to detect and deal with the error problems, if any, that arose during testing. The team might recommend additional training for users to reduce the errors as part of the implementation plan.

As noted, the role of the project team in testing and implementation will vary. In some cases, the team will manage or conduct much or all of the testing, training, conversion, and implementation. In other situations, a different group of people may perform part or all of the testing and, particularly, the implementation. The project team's role may consist primarily of preparing plans and documentation for the follow on groups. The rest of this chapter covers the preparation of testing and implementation plans.

TESTING PLANS

Testing addresses how well the proposed system achieves the following broad goals:

- The system logic performs as intended by the team.
- The system operates and performs in ways that meet or satisfy the requirements of the client.
- The system and documentation conform to the standards set forth by the client and organizational standards, including consistency with all other work prepared by the team.

Most **testing plans** strive to meet all of the aforementioned goals although occasionally the team may wish to focus on one or two when substantial further work by other people is required to finish and implement the system. As the team designs each part of the system, the team can ask, "How should we test to see that this part of the system meets the three goals?"

The test plan design begins during the project definition phase, and the team updates the plan as the project proceeds. The team and the client will carry out a number of test procedures as part of the project. The test plan states who is responsible for each test procedure or component and shows the deliverables. These test steps form part of the overall development plan for the project and should appear in the project Gantt chart or activity table. Points in the project at which the team should plan test activities include:

- **System specifications tests.** The team tests the conceptual specifications before investing effort in detail design specifications.
- **Design specifications tests.** Does the solution match the specifications?
- **Solution logic tests.** Does the solution perform as specified?
- **Solution value tests.** Does the system in operation meet the client and user requirements?
- **Report and documentation tests.** Does every report and document produced by the team meet team, client, and organizational standards? A bad report to management can damage the prospects for a good system.

The design team adds appropriate corrective action to the project plan for any testing issues that the team encounters. When, as frequently happens, the design team turns the system over to the client or another group for further work, the team notes in the test plan the actions the team has taken and the results. In the plan, the team also recommends additional test steps as appropriate for the new group to conduct.

The test plan may include desk checks, system walk-throughs, operational tests, documentation clearance, and other areas as appropriate. For example, the plan may suggest a live data test or perhaps a pilot test by a selected group of users. An example of a testing plan for GB Video is shown in Table 13.1. The possible content areas for the plan are addressed in the next sections.

Desk Checks

Desk checks form the first line of defense against system errors and problems. Every team member should conduct individual desk checks of his or her own work. The team member mentally reviews the work to determine if the work is consistent with other existing work by the team, contains correct logic, and meets the manager and client requirements. Normally the analyst will desk check his or her work at every significant milestone, including the completion of a major segment of code, data, process, or procedure. If several people have worked on a segment, the people who worked on the segment may join together for desk checks of the joint work. Desk checking should catch most if not all of the major errors within data and processes. Both correctness and completeness are important.

Walk-Through Tests

Walk-through tests form a second tier of testing activities. In a walk-through, the test participants as a group go step by step through the requirements as found in the narrative, data models, and process models checking at each step

TABLE 13.1
Testing Plan

Test Procedure	Deliverable	Responsible Person	Anticipated Date
Desk checks and walk-through	Project definition	Dan Cartperson and the team	Jan. 31
Desk checks and walk-through	Proposed system conceptual specifications test	Terrie Shaftkopf and the team	Feb. 25
Desk check and design specs walk-through	Solution logic test	Al Price and team	Mar. 10
Simulation walk-through test	Solution value test	Terrie Shaftkopf and the team	Mar. 12–13
Operational test	POC model initial tests	Dick Von Kemp	Mar. 30–Apr. 14
Operational test	Final POC test	Dick Von Kemp	Apr. 15
Desk checks and walk-throughs	Final report and presentation clearance	Team, manager and clients	Apr. 19–23
Production system operational test with live data	Acceptance test per the RFP	IT staff and vendor selection committee	Feb 20–Mar. 20 of next year
Desk checks and walk-through by IT and system users	Production documentation clearance	IT staff	Mar. 20–27 of next year
Production system operational test with actual users and customers	Post-implementation audit	IT staff and internal auditor	June 1–30 of next year

to see that the work is consistent and meets the goals set by the client. The test group may consist only of team members or of team members plus managers, clients, and users. The team may wish to assign one or more specific testing responsibilities to team members. For example, the team may ask one person to check the work prior to the walk-through to identify and make a list of any potential issues and questions. For each error or question identified during the walk-through, the team can assign the problem to a specific person for correction. If the errors were substantial, the team should schedule additional walk-throughs until all obvious errors are corrected.

The group may wish to conduct more elaborate walk-throughs that simulate the operation of the system. In a simulation walk-through, the team members play the role of system components: One person can play the role of a customer, another a system user, another the database engine, and others the processor. Following the specifications in the documentation, the group plays out what happens for some typical scenarios. As noted earlier, the team also may choose to develop a computer-based prototype and use the prototype to test logic, client and manager requirements during the walk-through.

When a team member or the team completes and has desk checked a section of the project, the team may wish to conduct **internal walk-throughs**, walk-through

tests conducted with only the team members in attendance. Normally, the group will find problems missed by the team member, especially problems of interaction, completeness, and consistency for the combination of work carried out by different team members.

When the team becomes confident that the solution is free of obvious errors, the next step involves **external walk-throughs**, which means reviewing the work with the manager and/or with client representatives and users. Taking work to the manager, clients, or users that contains obvious errors is inexcusable and may lead to serious negative consequences. Obvious problems suggest that the team either is unable to understand the standards and content for the project or that the members do not care enough to do it right. In other words, the team either ignored or put little effort into desk checks and internal walk-throughs.

The team may wish to begin with a review with the manager to whom the team reports. Once the manager is satisfied with the work, the team can schedule reviews with the client and users. Reviews with managers, clients, and users can pose complex issues. Both the determination and communication of requirements often are difficult for clients and users. In addition, clients and users may disagree on what they want. In an external walk-through, the team members present their work and encourage an open and thorough discussion with the manager, clients, and users. The team should expect at each review to find mistakes or learn about new issues. A good team avoids defensive behavior and encourages the clients and users to point out problems.

Design Specifications Walk-Through Tests

As noted previously, requirements and detail design may proceed sequentially, in parallel, or a mix. If sequential, the team conducts the same desk checks and walk-throughs for the detail design. The one new issue is checking the detail design against the requirements design to make sure the two match. The team continues to review the work with its manager and client. Once actual code or pseudocode exists, additional testing is possible. During desk checks or walk-throughs, the team can check the control flow to see that the trigger and conditional statements work correctly. Every module must either end the program, pause to await an external trigger, or trigger another module. Every module must be triggered by another module or by an external trigger. Every condition must lead to the correct action. Modules can be checked for functional and temporal cohesion and for possible data coupling problems.

Operational Testing

Once an operational solution, that is, a program or programs that execute, exists, the team can begin operation testing. Operation testing provides much more rigor than other forms because the computer actually performs the actions exactly as set forth in the program. This stage may reveal a number of problems that escaped detection in all the earlier testing. When the team plans to recommend purchase of a package solution, actual or demonstration packages may be available early in the project. These packages allow the team to perform some operational testing before any decision is made on how to proceed. When the

team plans to recommend building a solution, operational testing must wait until the system or an initial version or prototype is built and working.

The design team can perform operational tests to determine that the program compiles, performs actions as intended, and that the triggers and conditional logic work correctly. Much of this testing can be conducted before the program is complete. In place of the full logic for some modules, particularly the very complex ones, the design team can substitute a few simple statements or write a “The program reached Module P3.7 at (current time)” statement. This type of testing sometimes is called **stub testing**. Trace programs exist that will record and display the modules in the sequence as they execute and other parameters.

If the program logic and sequence control appear correct, the team can test the program with data. The team may wish to construct a test data set or the team may use live data. Often **live test data** for the system will exist in the form of a set of actual inputs with the corresponding outputs they produced. The team may have to write a program to reformat the inputs to use the live data. At some point the team may convert the existing files and load the data into the new system. Live data provides a good test but may not reveal problems with seldom used or newly added features. In another type of live data test, the team asks a group of users to exercise the system, by providing typical kinds of inputs and seeing if the system produces the expected outputs. Users, for example, order clerks, often know the kinds of issues that may cause problems for the system; however, their experiences come from the current system, not the new one. User tests also help the team to devise appropriate training procedures, manuals, and online help functions.

With **constructed data** the team can test seldomly used features of the system. Part of constructing a test data set is devising the outputs expected for the constructed inputs. Constructing anything other than a small data set can involve a lot of time and effort. Clients and users should participate in the creation of all test data. IT-oriented analysts and programmers may not know about all the conditions or may overlook problems that clients and users face every day.

Post-Implementation Tests

While careful testing of the system operation prior to implementation can provide much insight, some problems will appear only during post-implementation operations, usually at the most inconvenient time, such as on weekends, in the middle of the night, or during the busiest time of the year. A good test plan pays special attention to post-implementation testing and monitoring. Undetected problems in the post-implementation period can require very expensive corrective action.

Documentation Clearance

Documentation deserves the same complete and rigorous testing as all other aspects of a solution. The test plan should provide for an appropriate person or group to read, check, and certify every important piece of documentation. The tests include checks for:

- **Format.** The documentation should be free of errors that might impede understanding, such as missing words or sections, or pages out of sequence.

- **Correctness.** Specifications should accurately describe the system as built or in the final design. Instructions, data scheme, process diagrams, and metadata, when implemented, should produce the desired results.
- **Completeness.** The documentation should cover every important issue or action for successful operation and maintenance of the system.
- **Policy.** The actions, instructions, and implications of the documentation should follow or, at least, not conflict with organizational policies and culture.
- **Readability.** The intended users should understand the meaning of the documentation.

Documentation clearance often seems unglamorous and less important than other tasks. However, when the documentation is needed and used, errors and omissions can cause havoc. For this reason, the test plan should provide for explicit and careful tests of documentation. Many times, the first good test of documentation for completeness and accuracy occurs with the event of an emergency. If the documentation contains errors, serious consequences may result. The people who will use the documentation, and when possible, people who did not prepare it, should participate in the testing. The deliverable is a specific sign-off or clearance by a person or group for every piece of documentation. CASE tools, some programming languages, and database engines can provide good self-documentation when used properly.

IMPLEMENTATION

During implementation the proposed solution evolves into the operational or current solution. Implementation may involve moving toward the use of a modified existing system, a new built in-house system, or a purchased package or the transfer of activities to an outsourcer. Implementation may or may not involve changes to infrastructure and organization. As previously noted, the team may manage the implementation or other groups may take partial or complete responsibility for it.

The Implementation Plan

In all events, the project team needs to prepare an **implementation plan**. A typical implementation plan, may address the following topics:

- **Implementation strategy.** The general framework for implementation.
- **Integration requirements.** Preimplementation changes or modifications required to begin using the proposed system.
 - **Additional design.** Work needed before the new system starts operation including interfaces with other systems and correction of problems discovered during testing.
 - **Data.** Conversion of existing data files and input data formats to the new design.
 - **Infrastructure.** Converting the existing infrastructure to the facilities, software, and hardware required to operate the new system.

- Organization management. Who will manage the implementation and who will do the tasks needed to implement and operate the new system?
- Documentation. Most of the tasks during implementation and subsequent operation will rely on documentation provided or arranged for by the project team. The team needs to specify what documentation is provided and how to access it.
- System operations. Who will operate the system and under what rules or procedures?
 - People management. Hiring or assigning the people to operate and use the new system and training them. Training often is a major and expensive task.
 - Security management. Planning for backup and recovery, including specifying the people with responsibility and authority for procedural changes, system changes, data ownership, program and/or database changes.
 - Database administration. Managing the database with people and rules.
- Maintenance. Who will make the modifications that most systems require during the first months of operation to correct errors or omissions (and throughout the life of the system)? Who will modify the system to add features or respond to the changing environment and how this will change control work?

Data conversion includes all of the actions to allow the system to use, as required, existing files or databases and to accept inputs from and provide outputs to other systems. Teams may forget about data conversion until the time comes to run live tests. The functions of the conversion process may include the following:

- Prepare for new formats. Many times the analyst will need completely new formats for the I/O and the reports. This need may result from the use of a new database or a change in the input source.
- Provide data validation techniques. The team sets up the methodology to validate the data as converted to the new system.
- Specify new coding. The team may need to establish new coding structures for the old legacy data. For example, when the U.S. Postal Service changed the state abbreviations to a standardized two-digit format, systems using the old abbreviation of states such as *Alabama* as *Ala.* converted to the new code, AL.
- Design new forms. Most new systems will use new forms. The team should understand the impact of new forms and contribute to their design. As noted earlier, the content and format of the forms must match the process and data structure and allow the user to move through the form in a sequential path.

Implementation Strategies

As a first step in the implementation plan, the team selects an implementation strategy. The two broad choices are:

1. **Parallel implementation.** The new system operates in parallel with the existing system for a period of time. Both systems process the same inputs. The

client can compare the outputs produced by the two systems to determine if they agree.

2. **Direct Implementation.** The existing system is shut down at the time the new system begins operation.

With each of these two strategies, the team may select substrategies or sequential approaches known as *pilot implementation* and *phased implementation*. With **phased implementation**, the proposed solution is implemented one piece at a time. With **pilot implementation**, the new solution is implemented at one or several of many possible locations. The choice between the various strategies reflects an evaluation of the cost and risk that the team and client choose to accept.

Direct implementation

Clearly, the immediate costs of direct implementation are smaller than they are for parallel implementation because the costs of operating both systems simultaneously are eliminated. However, the total costs may be larger if the new system fails to operate correctly and causes such problems as data corruption that can be very costly to correct. Even worse, the new system may fail to perform with severe or even disastrous consequences to client, vendor, or customer relations; sales, and profits. When serious problems occur, top management, post facto, may view direct implementation as reckless or ill-advised regardless of how much thought and/or participation went into the decision.

The team normally selects direct implementation primarily for small or non-mission-critical systems—those systems with little or no effect on an organization's major performance measures. Direct implementation also may work well for a system where secure backup data exist. For example, organizations often use direct implementation for analysis and reporting systems that work off copies of operating databases because no risk exists of damaging important data. For small, noncritical or secure data applications, the extra costs of parallel implementation probably exceed any benefits. Direct implementation involves less risk when the team and client both have extensive experience with the content and technologies in the new system, for example, minor modifications to a stable, existing system by the team that built it. Direct implementation, when successful, allows the client to begin receiving the benefits of the new system right away.

As part of the direct implementation procedures, the team must provide the client with a viable backup or fallback option—a recovery plan in the event the new system does not perform at a satisfactory level. When a crisis occurs, people tend to make poor on-the-spot decisions. Making a backup plan requires the team and client to come up with a recovery plan in a noncrisis environment that can be invoked if and when a crisis occurs. When direct implementation is considered, the team also may wish to recommend much more extensive testing prior to implementation, a costly activity that further reduces the cost advantage of direct implementation. When prototype development is selected, both the client and the team can observe the prototype in a number of simulated or actual “live operations” often with actual users. These operations may provide enough

confidence in the new system for the team and client to agree on direct implementation. If a current system does not exist, then direct implementation may offer the only choice. The backup plan in this event may be simple: In a crisis, go back to whatever you were doing until the problems with the new system are corrected.

Parallel Implementation

Most large, mission-critical systems undergo some form of parallel implementation. When any major doubts exist about the wisdom of direct implementation, the team should look carefully at parallel implementation. In many cases, the potential extra costs of parallel implementation represent only a small part of the total cost of a new system. Even with extensive testing, most clients want to see the new system operate “online” for a while before authorizing the change or cut over to it as the sole provider of IT services for the function. Both clients and teams find parallel implementation less stressful than direct implementation.

How long to operate in parallel poses some difficult questions. For example, in many retail operations, the Christmas season represents a much heavier system load than the rest of the year. Typically, clients want to begin parallel operation when the load is light so people are available to address problems as they arise. However, some problems with the new system may only show up under high load conditions. Sometimes problems show up only when an unusual condition occurs, perhaps a major recall. In short, parallel operation increases confidence that the new system will perform as intended, but does not guarantee that problems will not arise in the future.

Sequential Approaches

As noted in earlier chapters, sequential or step-by-step approaches to development lower risk. Two sequential options may exist for implementation:

1. Phased implementation. Implement the new system one part or function at a time.
2. Pilot implementation. Implement the new system at one or several of a number of locations that eventually will use it.

Phased implementation is most useful with large systems that perform a number of functions. The idea is to implement the major functions one or several at a time. For example, consider the issues of replacing a large mainframe-based airline reservation system with a new client/server version. The risks associated with direct implementation are very large, and the costs of parallel implementation also are large. In this event, the team may elect to phase the implementation. For example, the first step might involve moving the customer database from a flat file on the mainframe to a relational database system on a separate server. This approach starts with a relatively simple function and has the benefit of freeing up time on the mainframe. The phased approach may incur significant costs to interface the new function with the existing system. In the example, the new customer relational database system may need a complex application program interface (API) to allow it to work with the rest of the mainframe reservation system.

TABLE 13.2
A Summary of
Implementation
Strategies

Option	Advantages	Disadvantages
Parallel—most appropriate for large or mission-critical systems	May reduce the risk of major operational problems. Less stressful for the client and team—i.e., may save your job	Incurs the extra expense of operating two systems. May not find all the problems. Delays realizing the benefits of the new system
Direct—most appropriate for smaller or non-mission-critical systems	Faster realization of benefits and less expensive when it works without major problems	Possible additional expense or a disaster when major problems occur. May appear “reckless” to top management
Sequential phased—most appropriate for complex multifunction systems	Lowers complexity and resource requirements and allows client and team to focus on one part at a time. May allow faster realization of some benefits	May involve significant extra expense to create APIs. Delays realizing the benefits of the full new system
Sequential pilot—most appropriate for systems that will operate at multiple locations	Lowers or limits negative impact on the overall organization of any problems that occur. Allows changes and revision to be made before widespread training occurs	Involves extra expense to manage the pilot. May require APIs. Delays realizing the benefits of the full new system

Subsequent steps might be to convert the reservation records and then the flight availability data to RDBs on servers. The last step might be to convert the very complex logic for finding and organizing the appropriate flights to respond to a customer request from the mainframe to a server. Note that the organization can implement the individual functions with either general implementation strategy—direct or parallel. With a parallel strategy, the cost is reduced because only part of the system is operated in parallel. With a direct strategy, the risk or potential cost of a major problem may be reduced compared to directly implementing the entire new system.

Pilot implementation is most useful when a system will operate at multiple locations, for example, the retail sale system for Wal-Mart or the inventory system for Air Force bases. Pilot implementation may work well for GB Video. The team can implement the new system at one store first and then implement at the other stores when all works well. As with phased implementation, pilot implementation can occur with a direct or parallel strategy. In both cases the cost and risks are reduced because the new system involves only a portion of the total volume or activity of the organization. Table 13.2 shows a summary of implementation strategies.

Training

Adequate training for all the people involved in a proposed system implementation, including users, operators, administrators, managers, and maintainers can exercise a critical impact on success. Breakdowns in training at any level can contribute to system failure at worst or to reduced performance at best. Adequate training will not ensure the success of a poorly designed system. But even with a good system, inadequate training can result in input, output, and system errors;

frustrate users and customers; antagonize managers; increase downtime and costs; and degrade performance. Training is expensive, but not training the relevant people can cost a lot more. With small projects, training responsibilities may be assigned to one of the team members. With large systems, training responsibilities often are assigned to a separate training group or organization.

Training goals in approximate priority order are to prepare the trainee to:

- Perform routine activities, for example, use, operation, or maintenance of the system.
- Handle promptly unusual events that may cause serious problems or damage, for example, a serious problem encountered by an angry customer or a virus attack on the system.
- Know how to obtain more information on unusual events that are less time-critical or that were not covered in other training.

Training may consist of initial and follow-on training both in a variety of formats. Individuals newly assigned to a role with a system normally receive some form of **initial training**. Initial training may range from a few minutes talk with a supervisor or co-worker to days, weeks, or months spent in formal training activities. Initial training should address all three of the training goals. Unfortunately, people forget, systems and conditions change, and unanticipated problems arise. **Follow-on training** and support mechanisms provide periodic or continuing training. Follow-on training may include such mechanisms as a help desk, online help, built-in help, and refresher courses.

Initial Training

This type of training tends to focus on direct users, the people in close and regular contact with the system. Initial training should answer the specific questions that the trainee will face, for example:

- How do I gain access to the system, log in, and so forth?
- How do I carry out my routine functions—rent videos or cars, take orders, answer questions, update data, and so on?
- How do I correct errors that I may make?
- How and under what conditions can I override the system, for example, change a price that the computer provides?
- What do I do when the unanticipated happens?

Format options for initial training include:

- Self-study.
- Informal conversations with a trainer, often a co-worker or manager.
- Computer-assisted instruction (CAI).
- Training sessions or courses.

The various options listed may utilize printed text and/or audio-visual materials. Many organizations now place training materials online. Online materials

can support initial training and also follow-on training. In common with all training materials, online materials quickly lose value unless they are updated as needed. However, the online format tends to facilitate the updating and dissemination of new materials.

Follow-on Training and Support

Even the best initial training provides only a beginning. People soon forget what they learned in initial training unless they use the information on a regular basis. Most systems undergo frequent modifications to correct problems or to add capability, and each change can lead to new training issues. While analysts and trainers try to cover a full range of situations, unanticipated problems always arise. People provide inputs and take actions that the designers never anticipated. Every system contains errors, some of which may not show up for years. In short, training and support are ongoing functions.

Refresher training resembles initial training. The organization arranges for people with substantial experience since initial training to undertake additional training. Refresher training may use any of the options and materials for initial training tailored to recognize the higher experience level of the trainees. Refresher training offers a good way to train for system modifications or changes in the work environment. Colleagues offer a major source for follow-on training. When a person encounters a situation that he or she cannot handle, the natural solution is to ask a co-worker or a supervisor. Initial training should encourage this approach. Some people will know a “system expert,” a person, perhaps in the IT organization, who knows the answers to a lot of the questions that arise. The demand on the time of the system expert may become so great as to interfere with the person’s primary duties.

Many organizations try to structure the on-demand dissemination of information from system experts. The daily support or **help desk** organization consists of a person or group of people with special tools and training to answer questions about the systems they support. A system user with a problem can telephone or e-mail the help desk for answers. A daily support group is an example of the “Train the Trainer” concept. The system developers provide guidance or prepare information for the intensive training of a small group of daily support people. The daily support people then train large numbers of system participants as issues arise. The support group needs to have a mechanism in place to acquire the necessary skills and information and to maintain and update the skills and information as the system and people change. The support organization may have a trainer assigned to the development team for the proposed system in order to acquire detailed support information as the project proceeds.

The training plan should identify whether or not a daily support group is recommended and, if recommended, how it will operate. In small organizations, the group may consist of one person with other duties. In large organizations, a multiperson group may exist. The support group working hours should match the times of operation of mission-critical functions. If the system, for example, a reservation system, functions 24×7 and the daily support group contains the only available “experts,” then the support group should function 24×7 . Often, the support

group sets up a Web site to relieve the load on the staff. The Web site contains answers and instructions for common tasks and problems. In many situations, the Web site may give the only help available outside of normal working hours.

The Training Plan

The training plan for a proposed system should cover both initial and follow-on training. System participants may exist at widely diverse skill and responsibility levels ranging from clerks entering data to scientists and engineers to senior managers. Direct users also may work at locations literally around the world, for example, checkout associates at Wal-Mart or reservation agents at Hertz. For these organizations and even for organizations that operate only domestically, training may involve language and culture differences. The training plan may tailor or segment the training for direct users with the following classifications in mind:

- Job responsibilities
- Organization level
- Educational level
- Language skills
- Cultural background

Each major combination of the above factors may lead to a different set of training materials and activities. Each trainee should receive training relevant to his or her role in the system and in a format that he or she understands and accepts.

The training plan specifies the training options and materials to be used. The team provides training materials unless the client directs otherwise. In small organizations, training materials probably will consist mostly of text instructions and guidance from colleagues. When a team purchases a package system, the vendor may provide printed and/or online text materials plus audio-visual and/or computer-assisted training aids. As noted, the team may or may not conduct training, but the plan should specify who will conduct the training. The plan also specifies how, when, and by whom these materials are used. The team may need to provide for a variety of special training activities in the training plan. For example, the people administering the organizational networks databases and servers that provide the infrastructure for the proposed system may need training on the operations and requirements. These people also may need training to understand the impact of the new system on the existing systems and other organizational units. System administrators need to understand how this new system fits with the organizational goals and policies.

Management may benefit from training on features, limitations, costs, and benefits including competitive advantages, if any, for the proposed system. Management may not want or need the detail provided to the direct participants; however, they may want information to make good investment decisions tied to strategic alignment in follow-on systems and to discuss competitive issues with investors and other managers. Table 13.3 gives an example of a training plan.

TABLE 13.3
Training Plan

Activity	Person Responsible	Completion Date
Training development	Al Trainor	Mar. 1
Initial training	Ted Teecher	Mar. 18
Follow-on training and support	Mary Occe	As needed for new employees
Refresher training	Mary Occe	As needed after conversion

Maintenance Plan

Fewer things are more certain in life than the fact the proposed system will require maintenance. Some of the common problems that occur include incorrect program logic, input data omissions or incorrect definitions, and missing error checks. Even when the programs for the proposed system are complete and correct, changes in sponsor, client, and user preferences; organizational goals and missions; laws and regulations; and the competitive environment lead to a demand for system changes. Surveys suggest that IT organizations spend a majority of their budget on maintenance of existing systems.

The **maintenance plan** specifies how the team recommends the client maintain the proposed system after it is implemented. A first consideration is who will maintain the system. Many times the development team moves on to other tasks and another group takes over maintenance. The organization may have an internal group maintain systems or the team may look at contracting with a third party for the maintenance. When the team recommends the purchase of a package system, the maintenance plan often recommends that the package vendor provide the maintenance. In small organizations, maintenance may drive the selection of a recommended system. A small organization may have little, if any, capability for maintenance and may choose to outsource to an application service provider or to purchase a package system with vendor-provided maintenance.

Regardless of the maintenance source, the plan should set up or specify a process to identify, review, control, and track changes to the system. Some changes are close to mandatory, for example, the correction of errors that prevent the system from performing its functions. Some others may be largely optional, for example, some users want to change the colors on the input screens. With package systems, many changes are made by the vendor. The client's decision is whether or not to install each new release. At a minimum, the plan should recommend that the client establish a change control process and hopefully, the plan can provide guidance on how the process will operate.

Good system documentation is a prerequisite to effective system maintenance. When a different group from the development team will maintain the system, documentation assumes an even more critical role. Documentation is discussed in the next section. An example of a maintenance plan for GB Video is shown in Table 13.4.

Documentation

The primary purposes of system documentation are to facilitate correct operation of the system and program maintenance and modification. Documentation

TABLE 13.4
Maintenance
Plan for GB
Video

Activity	Responsible Party
Follow-on activity	Jim Croates
Management-required changes	Al Price
Log errors from users	Logged-on Web site
Review errors	Al Price
Track changes	Al Price
Changes to code	Vendor
Changes in options and configuration	Al Price

also provides a base for training materials. In a sense, all programs have complete documentation: The code provides the most explicit possible documentation for the program. However, most people, even programmers, find other representations of the system easier to understand and follow. A critical part of the implementation plan is defining the documentation that the client wants or agrees to for key points in the implementation cycle. The plan also may specify documentation requirements and procedures for system changes post-implementation.

The system development plan should specify the kinds of documentation beginning with the System Definition phase of the project and continuing to and sometimes beyond the closing of the project. The best approach to documentation is generating it at each stage as the project goes along. Sometimes organizations, because of time pressures, decide to postpone preparing documentation until after the system is working correctly. History suggests that postponed documentation tends to never arrive. Once implementation occurs, issues other than documentation always seem more urgent. Documentation poses a familiar cost/benefit issue. Little or no documentation saves documentation costs but may cause large extra operating and maintenance costs.

For work performed in-house or under contract, the team has a number of options for documentation relating to media, format, and level of detail. Media may include text and graphical materials, comments in the program or with fourth-generation languages, the code itself. An organization may set up a number of format standards and conventions for documentation and may specify a desired level of detail. Each section of this book illustrates the authors' documentation standards in text descriptions and examples. For example, Chapter 7 "Learning from the Current Situation," uses a narrative, a first explosion data flow diagram, an enterprise data model, and associated metadata for documentation of the current operation.

With the purchase of package systems, the vendor makes most of the decisions on documentation. The vendors focus in on configuration and use of the product. The vendor often discourages changes in the code and sometimes declines to provide any documentation on the source code. When a vendor is involved, the implementation plan should specify the materials the vendor is to provide and the delivery times for the materials. Manuals that arrive a month after implementation is completed provide little or no help for initial training.

At a minimum, documentation should provide enough information to allow system participants to:

- Configure, load, and initiate the system.
- Format and load initial databases.
- Set data access and other security controls.
- Provide the needed input data in acceptable formats.
- Access all the desired system features.
- Recover in the event of a disaster or crash.
- Install new releases or updates.

The team reviews all documentation for clarity and accuracy. When the organization plans to maintain the system, the documentation should facilitate and track maintenance and modifications.

System Controls

Once a system is implemented and in operation, the best time to establish controls has passed. Good system controls must begin as part of implementation. The **system control** portion of the implementation plan specifies the who, what, when, and where for actions relating to the system. The plan may cover such areas as:

- Data access controls. Normally an access control list specifies who can create, retrieve, update, and delete data. Access may be place and time specific, for example, creates, updates, and deletions can occur only from a terminal directly connected to the system and only during normal working hours.
- Program or process initiation controls. Controls on causing a program or process within the program to execute.
- Database changes controls. Controls on changing the structure of the database, for example, adding a new attribute to a table.
- Program change controls. Controls on modifying programs.
- Physical infrastructure controls. Controls on access to and making changes in infrastructure.
- Organizational and procedure controls. Controls on changes to procedures that relate to the use of the system.

Disaster Plans

Disasters happen to the best of systems. The system may crash or, hopefully infrequently, be destroyed by fire, floods, tornadoes, disgruntled employees, terrorism, or other disasters. Crashes are the most likely occurrence. Some crash recovery features are built into the system; others are procedures that belong in the implementation plan. For example, in the event of a crash, MS Windows provides the last auto-saved version of the document if one exists. The user decides what to do next—replace the existing saved version of the document with the recovered (auto-saved) document or whatever else the user desires. Every system should have a plan for what to do after a crash and the plan may vary with circumstances. In a mission-critical activity, reverting to the last saved version

TABLE 13.5
GB Video
Disaster Plan

Activity	Responsible Party
Identify secondary storage devices with critical data	IT Director
Set up call list for emergencies	IT Director
Ensure that all data and programs are backed up on a daily basis	Database Administrator
Set up a cold off-site center	Facilities Director
Set up password system for clerks	IT Director

may result in the loss of a number of transactions or parts of transactions and often is unacceptable. More complex procedures are used in this situation.

Major disasters that cause physical loss or massive data loss happen infrequently, but can cause large expenses or organizational destruction. Organizations frequently provide remote storage of critical data and options or the use alternate physical facilities to protect from major disasters. An example of a **disaster plan** for GB Video is shown in Table 13.5. For an additional discussion of disaster plan recovery operations, see Weber, 1999.

Post-Implementation Audit Plan

Many times after a system goes into operation, an organization's managers give a sigh of relief and turn their attention elsewhere. Only when and if the system destroys a lot of data, misleads the organization, or annoys most of the customers do the managers check to see if the system is performing correctly. A **post-implementation audit** can detect problems before they become serious. In addition, an audit can provide valuable information for future projects. If acceptable to the client, the team should include a post-implementation audit in the implementation plan.

The post-implementation audit attempts to answer such questions as the following:

- How well does the system perform with respect to the strategic alignment goals?
- Does the system meet the stated system requirements or vendor specifications?
- Does the system have the proper controls in place?
- Were the system personnel properly trained?
- Does the system achieve the desired standard of usability?
- Does the system documentation conform to the organization's policy?

The team may include methods to collect data and an analysis process to answer the questions in the implementation plan. To avoid any suspicion of bias, however, the development team normally plays no role in the audit other than to provide background information.

GB Video Implementation Plan

The team in GB Video recommended that GB contact a vendor to custom build a video rental and return system. The vendor is expected to deliver the software about a year after the team completes the final report. The team's implementation plan in Figure 13.1 reflects these circumstances.

FIGURE 13.1 GB Video Implementation Plan

IMPLEMENTATION

The team prepared the following guidelines for implementation with the assumption that GB will contract with a vendor to build the new system. The GB staff can fill in the implementation materials once more is known about the product the vendor will provide.

Schedule of Implementation Activities

The team identified activities required to implement the new system in the table below. All dates refer to next year.

Activity	Responsible Person	Completion Date
Prepare a training plan	Vendor and GB staff	Mar. 1
Prepare a maintenance plan	Vendor and GB staff	Mar. 1
Prepare an operating plan	Vendor and GB staff	Mar. 1
Purchase hardware	GB staff	Jan. 5
Interface with existing systems	Vendor and GB staff	Apr. 1
Make organizational changes	Richard Cosier	Mar. 20
Conduct data conversion	Vendor	Feb. 1–Apr. 1
Train GB employees	Vendor and GB staff	Mar. 18– continuing
Install production system	Vendor and GB	Feb. 15
Conduct online testing	Vendor and GB	Feb. 20–Mar. 20
Pilot implementation	GB	Mar. 21
Begin full operation	GB	Apr. 1

Implementation Strategy

The team recommends that GB use a direct, pilot implementation strategy. On the morning of March 21, GB will implement the new system in the main store and discontinue the current system. GB will keep the forms and files from the current system until certain that the new system works. If serious problems arise, GB can go back to the current system with little cost and risk. Once the new system operates correctly, GB will implement the new system sequentially at the two remaining stores by the April 1 target date.

Other Implementation Activities

The team recommends that GB purchase the hardware several months before the new system test version is delivered. The vendor will suggest hardware and software that works well with the system as part of the RFP response.

The vendor contracted to carry out the required data conversion activities including converting the customer file and the video file. GB will need to make a decision about

rental transactions that are open on the date of conversion—enter into the new system or continue to use the old system for open rentals.

The team estimates that only three hours of training are required for clerks. With good design and help features, the system should be easy to use. The team recommends that GB not train the clerks until a day or so before the clerks start to use the system. The system operator(s) can receive training from the vendor.

The team understands that the vendor probably will maintain the system. GB also will need to make arrangements for hardware maintenance. Finally GB will wish to come up with an operating plan for the system.

Page 2

CLOSING THE PROJECT

Normally a manager or senior administrator makes the decision that a project is closed. A project may close because the work is complete, progress has stalled, the budget is exhausted, or other work has a higher priority. **Closing the project** consists of obtaining the manager's agreement that the team has completed all or as much as possible of the required work and tasks. At this point, the team members end their obligations to the project; however, they may still receive frantic calls for help up to years later. Many organizations use a project closing checklist that specifies the items for which the team must obtain sign-offs to close the project. A closing check list for a student field project might include:

- Having a manager sign off on the final report.
- Obtaining manager agreement that the team is ready to make a final presentation.
- Making the final presentation.
- Delivering the final report to the client complete with a program, or Web site for the working proof of concept model if the client so desires. Make sure the client has all of the instructions, passwords, and other items needed to run the model.
- Completing and submitting peer evaluations.

Summary

Completing a project means different things for each manager and client. Some clients want only the specifications to purchase or build a solution. Others want an operational solution, in other words, a system that meets their requirements installed and tested with good documentation. In all projects, the team should prepare testing and implementation plans and provide adequate documentation. In some cases the team will manage or conduct much or all of the testing, training,

conversion, and implementation. In other situations, a different group of people may perform part or all of the testing and, particularly, implementation.

Testing addresses how well the proposed system achieves three broad goals:

1. The system logic performs as intended by the designer.
2. The system transforms inputs to outputs in ways that meet or satisfy the requirements of the client.
3. The system and documentation conform to the standards agreed to with the project manager and client.

Points in the project at which the team should plan test activities include:

- System specifications tests.
- Design specifications tests.
- Solution logic tests.
- Solution value tests.
- Report and documentation tests.

The test plan may include desk checks, system walk-throughs, operational tests, documentation clearance, and other areas as appropriate. Every team member should conduct individual desk checks of his or her own work. The member should mentally review the work to determine if the work is consistent with other existing work by the team, contains correct logic, and meets the manager and client requirements. In a walk-through, the test participants as a group go step by step through the requirements as found in the narrative, data models, and process models; they check at each step to see that the work is consistent and meets the goals set by the client. When the team completes a section of the project, the team may wish to conduct internal walk-throughs, that is, walk-through tests conducted with only the team members in attendance. When the team gains confidence that the solution is free of obvious errors, a next step involves external walk-throughs, which involve reviewing the work with the manager and/or with client representatives and users.

Once an operational solution—a program or set of programs that execute—exists, the team can begin operational testing. Operational testing provides much more rigor than other forms because the computer actually performs the actions exactly as set forth in the program. When the team plans to recommend purchase of a package solution, actual or demonstration packages may be available early in the project. When the team plans to recommend building a solution, operational testing must wait until the system or an initial version or prototype is built and working. If the program logic appears correct, the team can test the program with data. Often live test data for the system will exist. Live test data are a set of actual inputs and the corresponding outputs. With constructed data the team can test seldomly used features of the system.

Documentation, including reports to clients and managers, deserves the same complete and rigorous testing as all other aspects of a solution. The test plan should provide for an appropriate person or group to read, check, and certify every important piece of documentation. The tests include checks for format, correctness, completeness, policy, and readability.

During implementation the proposed solution evolves into the operational or current solution. A typical implementation plan may address the following topics:

- Additional design work.
- Integration requirements. Pre-implementation changes or modifications to other systems and procedures required to interface with the proposed system.
- Implementation strategy. The general framework for implementation.
- Organization management. Who will manage the implementation and who will do the task needed to implement and operate the new system.
- Documentation. The team needs to specify what documentation is provided and how to access it.
- Data management. Conversion of existing data and database administration.
- Infrastructure management. Converting the existing infrastructure to the facilities, software, and hardware required to operate the new system.
- People management. Hiring or assigning the people to operate and use the new system and training them. Training often is a major and expensive task.
- System operations. Determine who will operate the system and under what rules or procedures.
- Maintenance. Determine who will make the modifications that most systems require.
- Security management. Plans for protection, backup, and recovery.

As a first step in the implementation plan, the team selects an implementation strategy. The two broad choices are:

1. Parallel implementation. The new system operates in parallel with the existing system for a period of time. Both systems process the same inputs. The client can compare the outputs produced by the two systems to determine if they agree.
2. Direct Implementation. The existing system is shut down at the time the new system begins operation.

With each of these two strategies, the team may select substrategies or sequential approaches known as pilot implementation and phased implementation. With phased implementation, the proposed solution is implemented one piece at a time. With pilot implementation, the new solution is implemented at one or several of many possible locations.

Adequate training for all the people involved in a proposed system implementation, including users, operators, administrators, managers, and maintainers, can exercise a critical impact on success. Training is expensive, but not training can cost a lot more. Training goals direct system participants in approximate priority order are to prepare the trainee to:

- Perform routine activities, for example, use, operation, or maintenance of the system.

- Promptly handle unusual events that may cause serious problems or damage, for example, a serious problem encountered by an angry customer or a virus attack on the system.
- Know how to obtain more information on unusual events that are less time-critical or that were not covered in other training.

Individuals newly assigned to a role with a system normally receive some form of initial training. Initial training may range from a few minutes of talking with a supervisor or co-worker to days, weeks, or months spent in formal training activities. Initial training should address all three of the training goals. Initial training tends to focus on direct users, that is, the people in close and regular contact with the system. Initial training should answer the specific questions that the trainee will face, for example:

- How do I gain access to the system?
- How do I carry out my routine functions?
- How do I correct errors that I may make?
- How and under what conditions can I override the system?
- What do I do when the unanticipated happens?

Follow-on training and support mechanisms provide periodic or continuing training. Follow-on training may include such mechanisms as a help desk, online help, built-in help, and refresher courses. People may forget what they learned in initial training; also, most systems undergo modifications and unanticipated problems always arise. Refresher training resembles initial training. The organization arranges for people with substantial experience since their initial training to undertake additional training. Colleagues offer a major source for follow-on training. The daily support or help desk organization consists of a person or group of people with special tools and training to answer questions about the systems they support. A system user with a problem can telephone or e-mail the help desk for answers.

The training plan for a proposed system should cover both initial and follow-on training. The training plan may tailor or segment the training with the following classifications in mind: job responsibilities, organization level, educational level, language skills, and cultural background. Format options for initial training include self study; informal conversations with a trainer—often a co-worker or manager; computer-assisted instruction (CAI); and training sessions or courses. The various options may utilize printed text and/or audio-visual materials. Online materials can support both initial training and follow-on training.

The maintenance plan specifies how the team recommends that the client maintain the proposed system. Many times the development team moves on to other tasks and another group takes over maintenance. The organization may have an internal group to maintain systems, or the team may look at contracting with a third party for the maintenance. When the team recommends the purchase of a package system, the maintenance plan often recommends that the package vendor provide the maintenance. Good documentation can facilitate maintenance. The implementation plan defines the documentation that the client wants or agrees to for key points in the implementation cycle.

The system control portion of the implementation plan specifies the who, what, when, and where for controls for data access, program or process initiation, database changes, program change, physical infrastructure changes, and changes to procedures. Every system should have a plan for what to do after a crash and other more extreme disasters. In a mission-critical activity, reverting to the last saved version may result in the loss of a number of transactions or parts of transactions and often is unacceptable. Organizations frequently provide remote storage of critical data and options on the use of alternate physical facilities to protect from major disasters. If acceptable to the client, the team should include a post-implementation audit in the implementation plan to answer questions about how the system actually performs.

Normally a manager or senior administrator makes the decision that a project is closed. Many organizations use a project closing checklist that specifies the items for which the team must obtain sign-offs to close the project. A closing check list for a student project might include sign-offs for:

- Manager approval on the final report.
- Manager approval for a final presentation.
- Team delivery of the final presentation to the client.
- Team delivery of the final report to the client.
- Team completion and submission of peer evaluations.

Key Terms

constructed data, 460	implementation plan, 461	project closing, 474
data conversion, 462	implementation strategy, 461	refresher training, 467
design specifications test, 457	initial training, 466	report and documentation test, 457
desk check, 456	internal walk-through, 458	solution logic test, 457
direct implementation, 463	live test data, 460	solution value test, 457
disaster plan, 472	maintenance plan, 469	stub testing, 460
documentation, 460	parallel implementation, 462	system control, 471
documentation clearance, 461	phased implementation, 463	system specification test, 457
external walk-through, 459	pilot implementation, 463	testing plan, 456
follow-on training, 466	post-implementation audit, 472	walk-through test, 457
help desk, 467		

Review Questions

Answer the following questions regarding these topics.

1. Testing plans.
 - a. What are the broad goals of a testing plan?
 - b. What should the test plan include?
2. Tests.
 - a. Describe desk checks, walk-through tests, and post-implementation tests. When is each one appropriate to use?

- b. What are the relative advantages of team-constructed test data and work sample data?
- c. Why should the users be involved with developing the test data?
- 3. Module design. Explain the following:
 - a. Data coupling problems.
 - b. Functional cohesion.
 - c. Temporal cohesion.
- 4. Implementation.
 - a. What are the critical components of an implementation plan?
 - b. List the advantages and disadvantages to parallel and direct implementation.
- 5. Implementation planning: What has to be done to prepare for each of the following during implementation?
 - a. Data conversion.
 - b. System installation.
 - c. Training.
 - d. Assure user acceptance.
- 6. Implementation strategies.
 - a. To install a new payroll program for your client, which implementation strategy would you likely use? Justify your answer.
 - b. To install a new Web site for your client, which implementation strategy would you use? Justify your answer.
- 7. Training. What is involved in:
 - a. Initial training?
 - b. Refresher training?
 - c. Ongoing training?
- 8. How would you tailor a training program based on students'
 - a. Educational level?
 - b. Job responsibilities?
 - c. Language skills?

Critical Thinking Exercises

Individual Exercises

1. Your client is a textbook publishing company. Your team is developing software to keep track of inventory, orders, and customer information. Prepare an implementation plan for the company.
2. You purchased a new computer. Develop an implementation plan for moving from your current machine to the new one.

Group Exercises

1. Prepare a comprehensive training plan for all aspects of Individual Exercise 1.
2. Prepare a comprehensive training plan for all aspects of Individual Exercise 2.

Reference

Weber, Ron. *Information Systems Control and Audit*. Upper Saddle River, NJ: Prentice Hall, 1999.

Appendix

GB Video Final Report

This appendix contains an example of a final report prepared for the GB Video Rental System Project. Many of the materials in the report come from examples in the chapters. The materials are arranged and integrated to give a full example of a final report that corresponds to the guidance in the book chapters. The report format follows the outline for a final report in Chapter 3 as shown below:

- Title Page
- Table of Contents
- Executive Summary
- Introduction
- Part I. Project Definition
- Part II. Proposed System Specifications
- Part III. Alternatives, Evaluation, and Recommendation
- Part IV. Design Specifications and/or RFP
- Part V. Implementation and Support
- Appendixes

Although the table of contents for the sample report does not display page numbers, page numbers normally appear in a table of contents. The table in the report shows first- and second-level headings. A team could display third-level headings if desired, but the first and second levels provide all the guidance a client needs.

A good team builds a final report as the project goes along. Most of the project definition materials go in with little change. The alternatives, evaluation, and recommendation material evolves as the project unfolds and the team learns more about the options. The team should continue to refine the report as the project proceeds. Team 7 made a few changes for the final report to the materials that appear in the earlier chapters. From a quality standpoint, the report probably ranks a little above average. Many student teams can produce better final reports.

A downloadable version of the final report in Microsoft Word appears on the book Web site (www.mhhe.com/vanhorn).

Final Report
GB Video Rental System Project

Prepared for:

Richard Cosier
President GB Video, Inc.

Prepared by
Team 7

Dan Cartperson
Dick Von Kemp
Al Price
Terrie Shaftkopf

April 23

TABLE OF CONTENTS

EXECUTIVE SUMMARY

INTRODUCTION

PART I. PROJECT DEFINITION

- Project Statement
- Strategic Alignment
- Project Success Criteria
- Features for the Proposed System
- Constraints
- Current Operations

PART II. PROPOSED SYSTEM SPECIFICATIONS

- Problem Solving
- Process Specifications
- Data Specifications
- Organizational Specifications

PART III. ALTERNATIVES, EVALUATION, AND RECOMMENDATION

- Alternatives
- Evaluation Comparison
- Recommendation

PART IV. REQUEST FOR PROPOSAL

- Qualified Bidders
- Evaluation and Selection

PART V. IMPLEMENTATION AND SUPPORT

- Implementation Schedule
- Testing Plan
- Implementation Strategy
- Hardware and Data Conversion
- Training
- Maintenance
- Emergency Plan

APPENDIXES

- Appendix A. Statement of Work
- Appendix B. Forms for the Current Situation
- Appendix C. Current System Data and Process Models
- Appendix D. Proposed Solution Data and Process Models
- Appendix E. Request for Proposal

EXECUTIVE SUMMARY

GB Video wishes to automate the current manual system for processing video rentals and returns. Reporting will be handled by a separate system. The new system should address improved customer service and lower handling costs for each transaction. GB asked Team 7 to do the analysis and design work necessary to acquire a new video rental system that will address the problems.

The team made every effort to align the project work with the strategic values of GB Video and President Cosier. The objectives table shows the expected impact of the proposed system features on the measures for the objectives that are important to President Cosier. The team identified features for the proposed system based on discussions with the client representatives. As shown in the table, these features contribute to the organizational goals identified in the strategic analysis—increased profits and sales.

Objective	Measure	Feature	Impact Target
1.1. Reduce the labor cost of renting and returning videos	Labor cost	Automate the rental and return processes	\$50,000 per year in labor cost savings
1.2. Benefit from economies of scale	Increase in revenues versus increased costs	Automated system handles increased volume with little or no new cost	Increases in costs equal to 85% or less of increases in revenues
1.4 Reduce the cost of nonreturned videos	Lost video costs	System must confirm member before renting	\$5,000 per year
2.2 Faster checkout	Checkout time	Automate process	Decrease from an average of 4.5 to 1.5 minutes
2.3 Tie video inventory to customer preferences	Increase in sales Increase in members	Rentals database available to Purchasing	5% increase a year
2.4 Targeted mail advertising to members	Increase in sales	Rentals and customer databases available for use	5% increase a year

Mr. Cosier stated that GB Video will consider any solution that will realize a payback in two years or less and cost less than \$200,000 up front to acquire and implement. The system should be in full operation in one year if possible. GB is willing to consider changes in function and organization if the changes provide significant benefits to GB. GB does not wish to continue with the current system unless all computer-based alternatives are prohibitively expensive, that is, outside the \$200,000 limit.

Mr. Cosier and his staff identified the following problems in the current situation:

- GB requires more clerks than similar stores with automated systems.
- Customers complain about long lines and slow checkout of rentals.
- Mistakes are common and some people use false member numbers and names to rent videos.
- Customers use late overdue notices as a reason to refuse to pay overdue charges.
- The marketing and purchasing departments lack the rental data that they need to do their jobs.

The team designed the specifications for the proposed system to alleviate all of the problems and to achieve the goals set by the client. Based on discussions with Mr. Cosier, the team identified four alternatives:

1. Improve the current manual system.
2. Procure a package system from a vendor.
3. Contract for service with an ASP.
4. Contract for a custom package.

Before performing in-depth analysis and evaluation of the four alternatives, the team conducted a feasibility study to determine if one or more of the alternatives failed to meet an important feasibility constraint. This analysis indicated that alternatives: 1. Improve the current manual system and 3. Contract for service with an ASP do not meet client constraints and should be dropped from further analysis. The team conducted an in-depth evaluation of alternatives 2. Procure a package system from a vendor and 4. Contract for a custom package. The team incorporated specifications for a custom package into an RFP and GB solicited bids. After reviewing the bids, the selection committee chose OkieComp, a local software company.

The features table summarizes the results of the evaluations of the alternatives. The features listed in the evaluation table are ones that the client mentioned or stressed.

Feature	Alternative			
	1. Improve	2. Package	3. ASP	4. Custom
Client preference	Low	High	Low	Possible
Improves performance	Some	Yes	Yes	Yes
Meets client constraints	No	Yes	No	Yes
Initial cost	\$5,000	\$65,000	\$500,000	\$99,500
Estimate payback	3 months	15 months	48 months	22 months
Meets specs	70%	95%	100%	100%
Client controls updates	Yes	No	Unclear	Yes
Risk	Low	Low	High	Medium
Time until operational	0	3 months	18 months	6 months
Custom modifications	Yes	No	No	Yes

All of the alternatives except the current system contribute to GB's performance objectives: they reduce cost by \$50,000, provide support for additional stores, and increase profits. The package and custom alternatives meet the constraints set by the client. The custom and ASP alternatives meet all of the design specifications.

The team recommends alternative 4. Contract for a custom package. After carefully reviewing the summary evaluation table, the team concludes that the custom package alternative does the best job of meeting the client's needs. The solution meets the constraints of a two-year or less payback and an initial investment of \$200,000 or less. The custom package contains all of the features desired by the client and thus does not require the client to change practices and procedures to fit the package. Both the team and Mr. Cosier concluded that the advantages of the custom package outweigh the disadvantages.

The activities required to implement the new system appear in the schedule table.

Activity	Responsible Person	Completion Date
Prepare a training plan	Vendor and GB staff	Mar. 1
Prepare a maintenance plan	Vendor and GB staff	Mar. 1
Prepare an operating plan	Vendor and GB staff	Mar. 1
Purchase hardware	Vendor	Jan. 5
Interface with existing systems	Vendor and GB staff	Apr. 1
Make organizational changes	Richard Cosier	Mar. 20
Conduct data conversion	Vendor	Feb. 1–Apr. 1
Train GB employees	Vendor and GB staff	Mar. 18– continuing
Install production system	Vendor and GB	Feb. 15
Conduct online testing	Vendor and GB	Feb. 20–Mar. 20
Pilot implementation	GB	Mar. 21
Begin full operation	GB	Apr. 1

The team provides the appropriate plans for implementation in the body of the final report.

INTRODUCTION

GB Video of Jackson, Oklahoma, asked Team 7 to perform the analysis and design work required for acquiring a new video rental system that will add value to the organization. Basically, GB Video wishes to automate the current manual system for processing video tape and DVD rentals and returns. A separate system will handle reporting

functions. The new system should address improved customer service and lower handling costs for each transaction.

This report contains the materials prepared by Team 7 for the clients. The report consists of five major parts and a number of appendixes. Part I. Project Definition examines the organization and the client's requirements for a new system. This part contains (1) the strategic alignment analysis for GB Video that identifies what the organization values; (2) features and constraints for the new system; and (3) a review of GB's current operation. The review identifies components and aspects of the current system to retain as well as the ones to change.

Part II. Proposed System Specifications builds upon the problem definition analysis to identify and refine the specifications for the proposed system. This part contains both narrative and graphical models of conceptual-level specifications for data and processing in the proposed system. Part III. Alternatives, Evaluation, and Recommendation examines alternative solutions for the client's problem. The section evaluates both build and buy options and makes a recommendation based on the evaluation.

Because the team and client decided to purchase a system, the team followed the purchase or RFP format option for Part IV. This part discusses the development of a request for proposal, the solicitation of bids, the evaluation of bid responses, and the selection of a vendor. Part V. Implementation and Support defines the work remaining to implement the new system for GB Video. Topics include the implementation schedule and strategy, training, maintenance, and other steps necessary to start the operation of the new system. The appendixes provide documentation and diagrams to supplement and detail the material in the report.

PART I. PROJECT DEFINITION

The team conducted an analysis of the GB Video's mission, vision, goals, and objectives to align the project and solution with the client's values. The team studied the organizational structure, the strategic alignment, and the goals and objectives of the organization's management team. The team evaluated the contribution of this project to the performance of the organization. The team worked with the client to identify the client's constraints and features for the proposed system and the client's success criteria for the project. The second part of this section contains descriptions of the current operations and the current physical and organizational infrastructure, an analysis of problems in the current operations and a determination of the aspects of the current situation to retain and the aspects to change.

Project Statement

GB Video wishes to automate their current manual system for processing videotape and DVD rentals and returns. Reporting will be handled by a separate system. The new system should address improved customer service and lower handling costs for each transaction. A discussion of activities the team agreed to undertake and client deliverables appears in the Statement of Work in Appendix A.

Strategic Alignment

The team worked closely with Mr. Cosier to determine the strategic alignment for the project. President Cosier has approved all statements on goals, objectives, and impacts of features.

Organization. Mr. Cosier, the owner and president of GB Video, stated, "The mission of GB Video is to serve the customer." The vision of GB Video is, "In the markets where GB owns stores, become the leading seller and renter of videos and related supplies with the best selection of videos at competitive prices and with better customer service than competitors." Mr. Cosier expressed concern that, in the future, national companies that rent videos from a Web site and mail them to customers may reduce GB's market. He hopes that the solution to current problems will offer GB the flexibility to compete with the national firms.

GB Video, with headquarters in Jackson, Oklahoma, operates three video stores in towns of about 10,000 people. Each store operates largely independently although the headquarters performs some functions for all three stores, for example, payroll, purchasing, and accounting. The company employs 37 people and realized revenues of \$1.5 million and profits of \$133,000 last year. The stores operate from 10 a.m. to 10 p.m. except on Sunday when the hours are 12 noon to 10 p.m. Each store has a store manager and the store managers report to Mr. Cosier. Within a store, two assistant managers and a number of full- and part-time clerks report to the store manager. The assistant managers are in charge of store operations on shifts when the manager is not on duty. The assistant managers and clerks perform any or all of the store functions as assigned by the manager on duty such as serve customers, shelve new videos, shelve returned videos, and send overdue reminders.

Goals and Objectives. Discussions with Mr. Cosier and his staff people identified the following strategic goals and objectives. GB wants to increase profit to 10 percent or more of revenue and to increase sales by 5 percent per year after adjusting for inflation. The profit increase will result from reducing the costs of renting and selling merchandise and from economies of scale as the business grows. Mr. Cosier is in the process of renegotiating the leases for his stores and expects to save money for current and future stores with lower rent. The 5 percent sales increase will result from offering customers fair prices at or slightly below those of competitors and providing faster checkout service and a wider range of selections than competitors. Mr. Cosier also wants to begin a targeted direct mail marketing effort using data about customers and rentals. The customer and rental data also will allow purchasing to do a better job of stocking the videos that are most in demand. He believes that these actions will increase the number of members, the number of visits per member, and the average number of videos rented on each customer visit.

The GB goals and objectives are summarized in Table A.1. Some of the information in the table comes for the proposed system features discussion in the next section of this report.

TABLE A.1 GB Video Goals and Objectives

GB Video Goals	Related Objectives
1. Increase GB Profitability to 10% or more of revenue	1.1. Reduce the labor cost of renting and returning videos
	1.2. Benefit from economies of scale
	1.3. Lower rent for stores
	1.4. Reduce cost of nonreturned videos rented to unknown nonmembers
2. Increase sales by 5% a year or more after inflation adjustments	2.1. Competitive prices
	2.2. Faster checkout
	2.3. Tie video inventory to customer preferences
	2.4. Targeted mail advertising to members

Impact on Performance. Mr. Cosier states that the new system is mission-critical for GB. Mr. Cosier plans to open additional stores if the new system does result in improvements. The new information system should increase profits by reducing labor cost per transaction and eliminating the cost of videos rented to nonmembers and not returned. Faster checkout service and better selections can lead to increased customer satisfaction, more members, and higher sales. GB looks at labor costs, profits, and revenues per store as major performance measures and at the total number of active members and the number of rentals per member. Table A.2 summarizes the impact of

TABLE A.2 Impact of Features of Objectives

Objective	Measure	Feature	Impact Target
1.1. Reduce the labor cost of renting and returning videos	Labor cost	Automate the rental and return processes	\$50,000 per year in labor cost savings
1.2. Benefit from economies of scale	Increase in revenues versus increased costs	Automated system handles increases in volume with little or no new cost	Increases in costs equal to 85% or less of increases in revenues
1.4. Reduce the cost of nonreturned videos	Lost video costs	System must confirm member before renting	\$5,000 per year
2.2. Faster checkout	Checkout time	Automate process	Decrease from an average of 4.5 to 1.5 minutes
2.3. Tie video inventory to customer preferences	Increase in sales; increase in members	Rentals database available to Purchasing	5% increase a year
2.4. Targeted mail advertising to members	Increase in sales	Rentals and customer databases available for use	5% increase a year

proposed system features on objectives. The impact target data in Table A.2 are derived from discussions between Mr. Cosier and the team and approved by him.

Project Success Criteria

Discussions with Mr. Cosier led to the following “measures of success” for the project. Mr. Cosier will consider the project a success if the team:

1. Completes the project on time. Since the team will not handle implementation, success is defined as submitting a complete final report by April 23rd.
2. Identifies and fully specifies a solution that provides the features desired by the client within the client’s constraints, or demonstrates clearly that no such solution exists. If no solution can be found, Mr. Cosier wishes to know this information as soon as possible so that he can consider modifying the requirements.

While Mr. Cosier feels strongly about the profit and sales goals, he understands that many factors other than the project influence profits and sales. He will consider the project a success if the team accomplishes items 1 and 2 above.

Features for the Proposed System

The team identified the following features based on discussions with the client representatives. As shown above, these features contribute to the organizational goals identified in the strategic analysis—increased profits and sales. The client wants the proposed rental and return system to include the following functions:

1. Membership
 - a. Collect data and store data on new members.
 - b. Update data for existing members.
 - c. Issue a member card to members.
2. Rental
 - a. Rent videos only to members.
 - b. Create and store a rental record with identification of member and video.
 - c. Adjust an inventory record to reflect the rental.
 - d. Issue a receipt to the customer/member.
3. Return
 - a. Update the rental record and the inventory record to reflect the return.
 - b. Calculate the overdue charge if any.

The client wants to change the processes above to require less clerk time.

The client also requests that the proposed system contain the following features:

1. Automate the entry of video ID and member number. The client asked the team to look at bar code scanning.

2. Automate the entry of the current date/time for rental and return.
3. Reduce the time the clerks spend working with the paper files—the files of return forms, videos, and members.
4. Eliminate duplicate manual data entry and storage, for example, entering the rental date and return date on both the video card and the rental form.
5. Eliminate the cost to manually send data to accounting.
6. Reduce the cost of separate credit card and rental transaction processing.
7. Move all reporting functions out of the rental and return system.
8. Automate the sending of overdue notices to customers as part of the rental system.
9. Provide rental data for a data warehouse to be used by purchasing.
10. Provide rental and customer data for a database for Marketing.

Constraints

Mr. Cosier states that GB will consider solutions that will realize a payback in two years or less and cost less than \$200,000 up front to acquire and implement. The system should be in full operation in one year if possible. GB is willing to consider changes in function and organization if the changes provide significant benefits to GB. In particular, GB wishes to transfer reporting and overdue notice functions to the accounting group. Adequate air-conditioned space in the headquarters exists for a server and network room. GB will acquire the equipment if that is the best alternative. Since GB currently owns no computers, interoperability with current hardware and software is not a problem.

GB does not wish to continue with the current system unless all computer-based alternatives are prohibitively expensive, that is, outside the \$200,000 limit. The procurement options that GB will consider include:

1. **Package System.** Search for a package system that will handle the rental and return tasks to the client's satisfaction. GB prefers to purchase a package as long as the costs to purchase and implement the package including infrastructure do not exceed \$200,000.
2. **Contract for Service.** Contract with an application service provider (ASP) to provide and operate the rental and return system. Mr. Cosier will consider contracting with an ASP if a suitable package is not available or cost-effective.
3. **Contract for a Custom Package.** Contract with an IT consulting company to create a custom package program for the rental and return system. Mr. Cosier will consider this alternative if a suitable package is not available or cost-effective.
4. **Build the System In-House.** Mr. Cosier states that GB Video has no in-house capability to build a system. Only if all other alternatives look unsatisfactory is he willing to consider developing such a capability. He believes that the cost probably is prohibitive.

Current Operations

In consonance with the request of Mr. Cosier, the team looked at video rental and return and closely related activities. GB Video operates in a manner similar to most video stores. GB rents only to customers who are members. If a customer wishes to become a member, GB will issue a membership provided the customer has a credit card, a telephone, and a government-issued picture ID such as a driver's license. The GB clerk obtains the name, address, credit card number, and expiration date from the picture ID and credit card, asks the customer for a telephone number, prints this data on a customer form, and assigns a unique member number. The clerk prepares a membership card with the name and member number and gives it to the customer. The customer form is placed in the customer file box. On every contact, the clerk asks the customer about changes and updates the customer form anytime a customer reports a change. Exhibit 1 in Appendix B shows a customer form.

Although most staff members at GB Video talk about renting "videotapes," GB actually rents more DVDs than tapes. This report uses the term *video* to refer to both tapes and DVDs. Customers may rent one or more videos for one or more days. The customer finds the desired videos and brings them to the counter. The clerk copies the name and number from the customer card and copies the title and unique ID number from the label on each video onto a prenumbered invoice form. Exhibit 2 in Appendix B shows an invoice form. If the customer forgets his or her member card, the clerk looks up the number in the customer file. The clerk also enters his or her employee number, the date of rental, and a due date for each video; computes the amount of the charge and tax; and gives a copy of the invoice to the customer as a receipt. Customers may pay for rentals with cash, check, or a credit card. The clerk notes the payment type on the invoice. For credit card payments, the clerk runs the customer's credit card through the credit card terminal and keys in the amount. The credit card company approves or denies the charge.

As time permits, a back office clerk processes the invoices. The date and rental number for each rental are entered on the file card for the video in the video rental file. The header data on the card for each video owned by the store shows the video number, title, vendor number, and date acquired. Purchasing enters this header data when a video is received. In this manner, the store can track the status of each video. Exhibit 3 in Appendix B shows a video rental card.

When a video is returned, a back office clerk retrieves the card for the video rental and the invoice, records the return date on the invoice and the video rental card, calculates overdue charges if any, and processes the credit card transaction. The credit card number is obtained by retrieving the customer form from the customer file. The overdue charges are noted on the invoice. The videos from a rental may be returned on different dates.

Once a week, the office uses the copies of the invoices and an employee file to prepare a report for the store manager showing the number of rentals for each video and for each clerk. The completed invoices and copies of video rental forms for videos that are more than three days overdue are sent to Accounting. As part of a separate

system, the accounting clerk uses the invoices to get revenue data for accounting records and sends out overdue notices to members as required.

The office also uses the video rental cards and the vendor file to prepare and mail a monthly report to each vendor showing the rentals for each video supplied by the vendor grouped by title. Graphical data and process models for the current operation appear in Appendix C.

Physical and Organizational Infrastructure. As noted in the description of current operations, GB Video operates a mostly manual system. The system uses paper forms and index cards in file boxes for data input, output, and storage. The clerks write on the forms and cards with pencil or pen. The credit card terminal is the only electronic communication device in the system. Each store owns and manages its own data. The store manager controls the operation of the store information system. The clerks and all other employees in the store report to the manager. Clerks are allowed to create and update invoice and customer records. Video rental records are created by Purchasing and updated by the clerks.

Problem Analysis. Mr. Cosier and his staff identified the following problems in the current situation:

- Because of the manual system, GB requires more clerks than similar stores with automated systems resulting in a higher cost and longer elapsed time per transaction.
- Customers complain about long lines and slow checkout of rentals. Some frustrated customers dump their videos on the counter and go off without completing the rental.
- Delays occur when several clerks wish to use the member or video card file at the same time.
- Mistakes are common. For example, the video card may indicate that the video is on the shelf—it shows a return date and no new rental date, but the actual video cannot be found. Clerks make mistakes in computing the charges. Customers often correct overcharges but remain silent when the clerk undercharges.
- People, including some members, use false member numbers and names to rent videos. If the customer does not show a member card, the clerk is supposed to check the member file. Because of the time and effort needed to check the file, the clerks omit checks most of the time especially when the store is busy. GB experiences a higher nonreturn or loss rate for videos than other similar stores.
- Accounting gets busy and does not send overdue notices to customers on a timely basis. Sometimes a video is several weeks' overdue before the customer receives a notice. Customers use the late notice as a reason to refuse to pay overdue charges.
- Because of the expense of preparing a manual mailing, GB does not do any direct mail marketing to members. Other stores have successfully used direct mail marketing to increase revenues.
- The rental data summarized by video is costly to prepare and of questionable accuracy. No data exist on customer preferences. Purchasing often uses guesses, estimates, and periodic direct observation to determine which videos and how many copies to buy.

Mr. Cosier estimates that correcting these problems should increase revenue by at least 5 percent and at the same time reduce costs by more than \$55,000 a year.

Retention and Change Analysis. The proposed system will retain, at the conceptual level, the functions in the current system for enrolling a member, renting a video, and returning a video. The proposed system will continue to collect all of the data collected by the current operation. However, the data model will change to one based on the “things” about which data are collected rather than on the forms and file cards in use. The proposed system may contain additional functionality to prevent nonmembers from renting videos.

The reporting functions will move to another system. The Rental and Return system will make the data it collects and stores available to the other systems. As a result of the move of the reporting functions and the new data model, a number of the data flows will change. Since the purchasing, reporting, and accounting functions will access the rental return data with their own systems, the Rental and Return system will show no flows to Accounting, Management, and Vendor. The contents of flows to and from the Invoice and Video Rental stores will change to match the new data model. The flows to and from the externals Customer and Credit Card Company remain unchanged in content.

The physical infrastructure will change from one based on manual operations to a computer-based infrastructure. The proposed system will have computer input and output devices that do not exist in the current system. The organization infrastructure will change in one respect—the clerks will interact with computer I/O devices. For the GB project, the analysis of the current situation provides a good base upon which to design the proposed system. With the structure outlined above, GB Video should realize its strategic goals of increases in profits and sales.

PART II. PROPOSED SYSTEM SPECIFICATIONS

The proposed computer-based Rental and Return system will focus on improved customer service and lower handling costs for each transaction. This narrative addresses the conceptual specifications for the proposed system. The specifications described in this model derive from the problems, features, and functions identified by GB Video during project definition. The team applied experience, difference reduction techniques, and best practices to arrive at the conceptual data and process specifications described in both narrative and graphical form in the following materials. The team recommends sourcing and evaluation in the next part.

Problem Solving

The proposed system builds on a modification of the current operation. The team members used their experiences with video rental systems plus information in the literature

TABLE A.3 Difference Reduction Applied to Problems

Problems in the Current Operation	Reduction Action or Operation to Arrive at a Problem-Free Proposed System (PS)
Incorrect rental charges	Include a calculation function in the PS
False or missing member numbers	Access the rental function only from the member function in the PS
Nontimely overdue notices	Include an overdue notice function in the PS

to arrive at a tentative set of features for the proposed system. The team incorporated the following best practices in the desired system:

- Capture rental and return input data only once. After the initial capture, retrieve the data from storage when they are needed.
- Single-point data storage of rental and customer data for use by the Rental and Return system and by Accounting.
- Automatic generation and insertion of rental number and customer number for a new member.
- System calculation of tax and total cost.
- System verification of customer data on each contact with a customer.

The team also used difference reduction. The difference reduction for problems appears in Table A.3 and for client-desired features in Table A.4. All of the features suggested by these tables are incorporated in the proposed system narrative and graphical models.

TABLE A.4 Difference Reduction Applied to Features

Current Operation	Client-Desired Proposed System (PS) Features	Reduction Action
Member, rental, and return functions	Faster, simpler, member, rental, and return functions	Redesign rental, return, and member functions
Duplicate data entry and data storage	Reduce the cost and errors of data entry	Include single-point entry and storage
Send data to accounting	Eliminate data transfer	Give accounting access to data stores
Credit card and rental form processing are entirely separate	Save the cost of separate processing of the rental form and the credit card	Integrate credit card processing into the rental function
Rental and return system contains two reporting functions	No reporting functions in the rental return system	Transfer reporting functions to Accounting

Process Specifications

The proposed GB Video Rental and Return system contains the following functions: (1) Member—create a new member record or update an existing one; (2) Rental—rent videos; (3) Return—return videos; and (4) Overdue—create overdue notices for videos that are overdue. These functions resemble the ones in the current operation. However, using the problem-solving methods previously described, the team introduced a number of changes to achieve the customer service and cost goals specified by the client. The client considers all of the functions described in the narrative mandatory.

The customers of GB Video identify the videos that they wish to rent and go to a check-out position. To improve the likelihood that GB rents only to customers who are members, the member option must be selected at the beginning of every rental transaction. The graphical process models in Appendix D provide further details on specifications.

Member. The member process is triggered by a customer request to (1) rent videos and/or (2) become a member. If the customer has and knows the customer number, the number is entered; the system retrieves the record from the Customer data store and displays the customer data. If the customer does not have the number, the customer can provide a telephone number (or a name and zip code). The system tries to retrieve the customer's record from the Customer data store. If the system is unable to retrieve the customer record or if the customer is not a member, the new member subprocess is initiated. The system will create a new member record provided the person has a credit card, telephone, and government-issued ID. The customer supplies the customer data and the data are entered. The system generates a customer number, creates a membership card output, and gives the output to the new member. The system creates a record in the Customer data store for the new member.

Once the appropriate customer record is available, the customer is shown and asked (1) to verify the name, address, telephone, and credit card data and (2) to report any changes or corrections. This subprocess increases the likelihood that the system will contain current information for the customer. Any change data are entered and the customer data store is updated. When a verified customer record is available and the customer wishes to rent, the member process triggers the rental option and the member data are sent to the rental process. The rental process can be accessed only from the member process; the rental process cannot be accessed directly.

Rental. The rental process is triggered by and only by the member process. The rental process accepts the member data from the member process and generates a new rental transaction number and the rental date. The customer provides the video number and the proposed return date, which is the due date. The video number and the due date are entered into the system. The system retrieves the video data from the Video data store and, based on the due date, calculates the rental price for each tape or DVD rental. This process is repeated for each video. After all the videos are processed, the system adds the rental price for each video to get a rental subtotal. The system

multiplies the rental subtotal by the tax rate (state sales tax + county sales tax + city sales tax) to get the tax. The total rental cost is the rental subtotal plus the tax.

The payment type—cash, check, or credit card—is supplied by the customer and entered into the system. If payment is by credit card, the credit card data are entered. The system sends the credit card data and the total rental cost to the credit card processor. If the transaction is approved (or is for cash or a check), the system “commits” the rental and line records; in other words, the records are created in the data stores. The rental number, date, clerk number, pay type, and credit card data go to the rental record. The due dates go to the line records for each video. The system then generates a receipt for the customer. The receipt data includes all of the data supplied by the customer, sent from the member process, and generated by the system during the rental transaction. The system also retrieves the title data from the Title data store and includes the name in the receipt data.

Return. When a video is returned, the video number is entered into the system. The system retrieves the corresponding line and rental records, enters the return date, and calculates overdue charges if any. The customer may charge the overdue fee to the credit card used for the rental or may pay by cash or check. About 95 percent of the time, customers return videos with no payment-type input, for example, drop them in a return box. In the absence of customer input on payment, the system retrieves a credit card number from the customer record and processes the overdue charge against the credit card. The system records the charge and payment type in the Line data store. The system may create a return receipt for the customer.

Overdue. After the overnight returns are processed, a clerk instructs the system to run the overdue program, that is, the clerk triggers the overdue function. The system processes the Line records. For each video that is two or more days overdue (i.e., today's date – due date ≥ 2), the system retrieves the rental record for the video, retrieves the customer record for the rental, generates an overdue notice, and sends the notice to the customer. When a video is 14 days overdue, the system retrieves the customer's credit card number from the Customer data store and the video cost from the Video and Title data stores, charges the customer's credit card for the amount of the cost of the video, and sends a notice informing the customer of the charge. If a customer has a complaint about overdue charges, the complaint is handled and processed by Accounting.

Data Specifications

The new computerized system will contain data about Customer, Rental, Line, Video, and Title. The conceptual data model in Appendix D shows a graphical representation of the data structure. The Customer entity will contain customer number, name, address, telephone number, credit card number, and expiration date. If a member does not rent a tape for 24 months, the record is deleted and the person must rejoin as a new member to rent videos. GB expects to start with about 4,000 members and this number will grow over time.

The Rental data will contain rental number, rental date, clerk number, payment type (cash, credit card, or check), and for credit card payments credit card number, expiration date, and the credit card approval code received from the credit card company. About 80 percent of rental payments consist of cash, 10 percent checks, and 10 percent credit cards. The Line data for each video rented will contain line number, due date, return date, overdue charge if any, and payment type for the overdue charge. An average rental consists of two videos, which results in two lines. Some rentals may consist of 10 or more videos. At the end of each week, the Accounting system will process the Rental and Line data, will transfer the data on completed rentals to a data warehouse, and will delete the records for completed rentals. On average, the rental file is expected to contain about 800 rental records just before processing by Accounting.

The Video data contain the rental fees for each video number. A video may have three fees: the one-day rental fee, a lower fee for additional days, and a special fee for a weekend and or/holiday. Since GB closes on Sundays and holidays, the special fee encourages members to rent several videos for Sunday and holiday periods. The Title data contain the title number, the name of the video, the vendor code, and the cost paid by GB to acquire the video. Title and Video define data stores that are accessed by the Rental and Return system but are created and maintained outside of the system.

Organizational Specifications

The organizational sources and destinations for external data are shown in the Context-Level DFD in Appendix D. The sales clerks, who report to each store manager, have authority to retrieve, create, or update records in the Member, Rental, and Line data stores and to retrieve records in the Video and Title stores. The Accounting department at headquarters has authority to retrieve records from all stores and to delete records in the Customer, Rental, and Line stores. Accounting produces all reports on rental activities. Records in the Title and Video data stores are created, updated, and deleted by Purchasing at headquarters.

PART III. ALTERNATIVES, EVALUATION, AND RECOMMENDATION

This section reviews the alternative approaches for acquiring a system that meets the needs of GB Video and evaluates the feasibility and cost-effectiveness of each one. Based on this information, the team compares the alternatives and makes a recommendation which Mr. Cosier accepted.

Alternatives

Based on discussions with Mr. Cosier, the team identified the set of alternatives shown below. The team has defined and evaluated each of the alternatives. Before performing in-depth analysis and evaluation of the four alternatives, the team conducted a feasibility study to determine if one or more of the alternatives fail to meet an important feasibility constraint. This analysis indicated that Alternative 1. Improve the Current Manual

System and Alternative 3. Contract for Service do not meet constraints and should be dropped from further analysis for reasons discussed below.

The team conducted an in-depth evaluation of Alternative 2. Procure a Package System from a Vendor and Alternative 4. Contract for a Custom Package. Because Mr. Cosier provided estimates of costs and benefits, the team applied cost/benefit analysis, identified advantages and disadvantages, and examined the level of risk. In accord with Mr. Cosier's preferences, the team calculated the payback period for these two alternatives.

Alternative 1. Improve the Current Manual System. Based on a preliminary review, the team believes that several low-cost changes (probably less than \$5,000) to the current manual system could bring immediate improvement at GB Video. The team estimates that these changes would achieve about 40 percent of the benefits mentioned by Mr. Cosier or a cost reduction of about \$20,000 a year giving a payback period of three months. The risk associated with this option is low. The primary risk is that the changes may not result in the expected cost savings, but even if the savings do not happen, the initial cost is less than \$5,000. When this option was discussed with Mr. Cosier, he repeated his statement that he does not wish to continue with the current system unless all computer-based alternatives are outside his payback criteria and his \$200,000 up-front cost limit. Since the team found other alternatives that will provide the desired functionality within the constraints, the team did not given further consideration to improving the current manual system.

Alternative 2. Procure a Package System from a Vendor. Package systems are software or computer program packages provided by third-party vendors. The team identified four vendors that sell or lease packages for video rental and return activities. After preliminary examination of specifications and demonstrations provided on the vendor Web sites, all four packages appear to meet the features requested by the client and the specifications derived by the team. The packages run under the NT operating system on any Intel chip server. The packages will run with MS Access, MS SQL server, or Oracle Server databases. The vendors claim that the packages can be installed and in operation in three months. Advantages of package systems for GB Video include the following:

- They meet most of the proposed system conceptual specifications.
- They are available for implementation in less than a month.
- They are relatively inexpensive and within the client's total investment constraint of \$200,000.
- They are tested, and have proven functionality and performance.
- They have relatively low risk.

Disadvantages include the following:

- They may require some organizational changes.
- They are dependent on the vendor for maintenance and upgrades.
- The fixed yearly cost for upgrades are independent of whether or not GB wants or needs the upgrade.

TABLE A.5 Cost Benefit Summary for Purchase of a Package

Costs	Initial	Year 1	Year 2	Year 3
Computer hardware	\$40,000	0	0	0
Package cost	\$15,000	0	0	0
Data and training	\$10,000	0	0	0
Hardware maintenance	0	\$1,000	\$1,000	\$1,000
Package support	0	\$2,000	\$2,000	\$2,000
Benefits				
Cost reduction	0	\$50,000	\$50,000	\$50,000
Profit increase	0	\$6,650	\$6,650	\$6,650
Net benefit	(\$65,000)	\$53,650	\$53,650	\$53,650
Cumulative net benefit	(\$65,000)	(\$11,350)	\$42,300	\$95,950

The team analyzed the costs associated with this alternative. Package costs appear to run around \$15,000 for initial purchase with use at the three GB Video existing stores. The vendors provide yearly maintenance and upgrades at around \$2,000 a year. The team estimates that installation, including data conversion and training, will cost \$10,000 and that hardware purchase costs for the three stores will total \$40,000. Hardware maintenance is estimated at \$1,000 a year.

The team examined the possible cost reductions from an automated system and concluded that Mr. Cosier's estimate of \$50,000 a year appears reasonable. The possible benefit of a 5 percent increase in sales is more difficult to analyze. Sales should increase by at least 5 percent for the existing stores and should lead to a more than 5 percent increase in profits. If Mr. Cosier opens additional stores as planned, the profits may increase by a significantly larger amount. To be conservative, the team used a 5 percent increase in profits for the existing stores or \$6,650 ($\$133,000 \times .05$). Table A.5 shows a summary of costs and benefits for purchase of a package.

If everything goes as planned, GB will recover its investment early in year 2 for a payback period of about 15 months ($12 + [11350/53650] \times 12 = 14.5$ months). Even if profits do not increase at all, the payback period of 17 months still meets the two-year payback constraint set by Mr. Cosier.

Alternative 3. Contract for Service. The team searched for possible ASP vendors who might submit a bid to perform the functions identified in the conceptual specifications for the GB Video system. The team contacted a selection of current ASP vendors and also talked with several large video rental companies to find possible bidders. The video companies declined to bid. One said that providing such a service to help a potential competitor was not in their best interests.

None of the ASP vendors currently provide a video rental system and none expressed any interest in buying one of the available video packages and gearing up to offer such a service. Several said that they did not think a viable market existed. The team did find one ASP, Integrated Computer Services (ICS), that would develop a video rental package and supply the service to GB Video via an Internet interface. ICS estimates that developing, testing, and installing the program at GB will take 18 months. ICS proposes that GB Video pay for the initial development, setup, and marketing of the service, estimated at \$500,000 and then share in the profits of subsequent sales of service to other video rental companies. ICS estimated that GB Video would recover its investment in four years and could make significant profits after that time. Mr. Cosier rejected this option as outside of the business plan for GB Video, incurring high risk, and providing an unacceptable payback period.

Alternative 4. Contract for a Custom Package. As discussed in the next part, the team incorporated specifications for a custom package in an RFP. GB sent the RFP to six potential contractors or bidders. After reviewing the bids, the selection committee chose OkieComp, a local software company. OkieComp submitted a bid to write a software package to detail design specifications provided by GB Video. OkieComp currently sells a similar well-regarded package developed for equipment rental stores. OkieComp stressed that the company provides programming services, not system design services. GB must give OkieComp detailed design specifications for the program and specify the infrastructure in which the programs will operate. OkieComp will write the software and deliver an extensively tested package to GB Video. OkieComp quoted a total initial cost of \$99,500 to develop, test, convert data, and install the system including hardware. OkieComp's bid contained the following conditions:

1. The initial cost for developing, testing, and installing the software and hardware that fully meets the specifications provided by GB Video in the RFP is \$94,500. The hardware list appears in Appendix A of the bid. The amount of \$94,500 will be paid at the time that GB accepts the software.
2. OkieComp will deliver the software six months after receiving a firm contract. OkieComp will work with GB Video to help GB Video demonstrate that the software works according to specs. OkieComp will correct any noncompliance issues identified by GB at no cost for five years.
3. If GB Video wishes to make any changes to the software or infrastructure, OkieComp will charge GB the actual labor cost to OkieComp plus 70 percent to make, test, and install changes to the software.
4. OkieComp will load the GB data into the new system and train the people at GB for a cost of \$5,000.

Advantages of OkieComp include the following:

- The product will perform exactly as specified by the team and client.
- No organizational changes are required.
- No payment is due until the product meets acceptance tests. This provision reduces the risk of damage to GB if OkieComp is unable to deliver.

TABLE A.6 Costs and Benefits for the OkieComp Proposal

Costs	Initial	Year 1	Year 2	Year 3
Package and hardware	\$94,500	0	0	0
Data and training	\$5,000	0	0	0
Hardware maintenance	0	\$1,000	\$1,000	\$1,000
Package support	0	0	0	0
Benefits				
Cost reduction	0	\$50,000	\$50,000	\$50,000
Profit increase	0	\$6,650	\$6,650	\$6,650
Net benefit	(\$99,500)	\$55,650	\$55,650	\$55,650
Cumulative Net Benefit	(\$99,500)	(\$43,850)	\$12,200	\$67,850

- GB controls the upgrades.
- There is a five-year guarantee.

Disadvantages of OkieComp include the following:

- A possible risk exists that OkieComp cannot deliver. This risk is estimated as small and GB has some protection.
- A possible risk exists that OkieComp will go out of business. This risk is estimated as medium.
- Upgrade costs are not fixed.
- Three months or more are required to obtain the software.

The costs for this alternative appear in the OkieComp proposal. As in the package system alternative, GB can obtain hardware maintenance from a third party for \$1,000 a year, and the benefits remain a cost reduction of \$50,000 and additional profits of \$6,650 per year. Table A.6 shows a summary for the OkieComp alternative.

If everything goes as planned, GB will recover its investment late in year 2 for a payback period of about 22 months ($12 + [43850/55650] * 12 = 21.5$ months). This alternative appears to meet the two-year payback constraints set by Mr. Cosier, but it probably involves more risk than the package alternative. If profits do not increase, the payback period is slightly more than two years. If OkieComp is unable to deliver on time or at all, the risk to GB is small. GB can continue to use the existing manual system and can purchase a package system if needed.

Evaluation Comparison

The team defined and evaluated four alternatives for the GB Video Rental System:

1. Improve the Current Manual System.
2. Procure a Package System from a Vendor.
3. Contract for Service.
4. Contract for a Custom Package.

TABLE A.7 Evaluation Summary Table for GB Video

Feature	Alternative			
	1. Improve	2. Package	3. Contract	4. Custom
Client preference	Low	High	Low	Possible
Improves performance	Some	Yes	Yes	Yes
Meets client constraints	No	Yes	No	Yes
Initial cost	\$5,000	\$65,000	\$500,000	\$99,500
Estimate payback	3 months	15 months	48 months	22 months
Meets specs	70%	95%	100%	100%
Client controls updates	Yes	No	Unclear	Yes
Risk	Low	Low	High	Medium
Time until operational	0	3 months	18 months	6 months
Custom modifications	Yes	No	No	Yes

Table A.7 summarizes the results of the evaluations of these alternatives. The features listed in the evaluation table are ones that the client mentioned or stressed.

As noted earlier, the client and team agreed not to explore alternatives 1 and 3 in depth. The data on them is presented here to provide perspective. All of the alternatives except the current system contribute to GB's performance objectives, that is, they reduce cost by \$50,000, provide support for additional stores, and increase profits. Alternatives 2 and 4 meet the constraints set forth by the client. Alternative 4 is the only alternative that meets all of the client's desired features.

Recommendation

The team recommends Alternative 4: Contract for a Custom Package. After carefully reviewing the summary evaluation table, the team concludes that the custom package alternative does the best job of meeting the client's needs. The solution meets the constraints of a two-year or less payback and an initial investment of \$200,000 or less. While the custom package has a longer payback period and higher initial cost than the off-the-shelf packages, it offers a number of advantages. It contains 100 percent of the features desired by the client and thus does not require the client to change practices and procedures to fit the package. The vendor also agrees to modify the package when and as requested by the client.

The risk of this alternative is limited by the provision that GB does not pay for the package until it is tested and accepted as operational. If, for some reason, OkieComp fails to deliver, GB can purchase an off-the-shelf package and have it operational in three months. The major risk to GB is that OkieComp may go out of business and not correct problems or make modifications if desired. During an informal review, both the team and

TABLE A.8 Bid Summary

Bid Responses	Adv. Software	OkieComp
Met or exceeded minimum qualifications	Yes	Yes
Meet all mandatory and desired specs	Yes	Yes
Initial software cost	\$109,950	\$94,500
Cost to load data into the new system	\$4,700 (with a subcontractor)	\$5,000
Cost for upgrades and modifications	Labor + 80%	Labor + 70%
Experience with similar projects	Good	Very good
Customer responses on delivery performance	Very good	Excellent
Resources available for the project	Adequate	Excellent

Mr. Cosier concluded that the risk is acceptable and that the advantages of the custom package outweigh the disadvantages.

PART IV. REQUEST FOR PROPOSAL

This section describes the work done to prepare a request for proposal, to solicit responses, and to evaluate the responses. The section describes the responses and provides a recommended product. The text of the RFP appears in Appendix E.

Qualified Bidders

As instructed by Mr. Cosier, the team prepared an RFP and sent it to six potential bidders that the team identified. Four bidders submitted proposals. Two of the four bids were deemed non-qualified and were eliminated from the analysis. The team prepared the bid summary in Table A.8 for the two finalists.

Evaluation and Selection

An evaluation committee of two team members, the RFP consultant, President Cosier, and Purchasing Director Olijer read the bids and assigned rankings to the evaluation criteria in the RFP. The resulting evaluation by the committee appears in Table A.9.

TABLE A.9 Bid Evaluation Using Weighted, Ranked Features

Features	Weight	Advanced Software		OkieComp, Inc.	
		Ranking	Weighted	Ranking	Weighted
Specifications	0.3	10	3.0	10	3.0
Experience	0.3	8	2.4	9	2.7
Deadlines	0.1	9	0.9	10	1.0
Capacity	0.2	8	1.6	10	2.0
Cost	0.1	7	0.7	10	1.0
Total	1.0	42	8.6	49	9.7

The evaluation committee recommends that GB enter into a contract with OkieComp, Inc. Both vendors are well qualified and submitted good bids. Although the differences are relatively modest, OkieComp received the highest raw and weighted scores and is the vendor preferred by Ms. Olijer. Conversations with the proposed project manager from OkieComp convinced the team that the company understands well both GB Video and the video rental industry. While both bidders have been in business for more than 10 years, the high failure rate of software firms poses some risk for future support from both bidders; the committee could not identify any risk differences between the two firms. If for some reason, GB is unable to reach a contract agreement with OkieComp, the committee finds the Advanced Software bid an acceptable alternative.

PART V. IMPLEMENTATION AND SUPPORT

This section contains implementation and support plans for the production rental system that OkieComp will deliver in about one year. The team prepared guidelines for implementation. Team members and/or GB staff will need to fill in the implementation materials once more is known about the product that OkieComp will provide.

Implementation Schedule

The activities required to implement the new system appear in Table A.10. All of the dates in the table refer to next year.

Testing Plan

The testing plan appears in Table A.11. The team completed all of the testing activities through April of this year. The team recommends additional testing as shown after OkieComp delivers the production programs and documentation.

TABLE A.10 GB Video Rental System Implementation Schedule

Activity	Responsible Person	Completion Date
Prepare a training plan	Vendor and GB staff	Mar. 1
Prepare a maintenance plan	Vendor and GB staff	Mar. 1
Prepare an operating plan	Vendor and GB staff	Mar. 1
Purchase hardware	Vendor	Jan. 5
Interface with existing systems	Vendor and GB staff	Apr. 1
Make organizational changes	Richard Cosier	Mar. 20
Conduct data conversion	Vendor	Feb. 1–Apr. 1
Train GB employees	Vendor and GB staff	Mar. 18–continuing
Install production system	Vendor and GB	Feb. 15
Conduct online testing	Vendor and GB	Feb. 20–Mar. 20
Pilot implementation	GB	Mar. 21
Begin full operation	GB	Apr. 1

TABLE A.11 Testing Plan

Test Procedure	Deliverable	Responsible Person	Anticipated Date
Desk checks and walk-through	Project definition	Dan Cartperson and the team	Jan. 31
Desk checks and walk-through	Proposed system conceptual specifications test	Terrie Shaftkopf and the team	Feb. 25
Desk check and design specs walk-through	Solution logic test	Al Price and the team	Mar. 10
Simulation walk-through test	Solution value test	Terrie Shaftkopf and the team	Mar. 12–13
Operational test	POC model initial tests	Dick Von Kemp	Mar. 30–Apr. 14
Operational test	Final POC test	Dick Von Kemp	Apr. 15
Desk checks and walk-throughs	Final report and presentation clearance	Team, manager, and clients	Apr. 19–23
Production system operational test with live data	Acceptance test per the RFP	IT staff and vendor selection committee	Feb 20–Mar. 20 of next year
Desk checks and walk-through by IT and system users	Production documentation clearance	IT staff	Mar. 20–27 of next year
Production system operational test with actual users and customers	Post-implementation audit	IT staff and internal auditor	June 1–30 of next year

Implementation Strategy

The team recommends that GB use a direct, pilot implementation strategy. On the morning of March 21, GB will implement the new system at the main store and discontinue the current system. GB will keep the forms and files from the current system until company managers are certain that the new system works. If serious problems arise, GB can go back to the current system with little cost and risk. Once the new system operates correctly, GB will implement the new system sequentially at the two remaining stores by the April 1 target date.

Hardware and Data Conversion

As part of the bid, OkieComp will purchase and install the hardware and convert the current data to the new system. The team recommends that GB ask OkieComp to purchase the hardware several months before the new system test version is delivered in order to allow time to install and test the hardware prior to any production software testing. Although the vendor will carry out the required data

TABLE A.12 Training Plan

Activity	Person Responsible	Completion Date
Training development	AI Trainor	Mar. 1
Initial training	Ted Teecher	Mar. 18
Follow-on training and support	Mary Occe	As needed for new employees
Refresher training	Mary Occe	As needed

conversion activities including converting the customer file and the video file, GB will need to make a decision about rental transactions that are open on the date of conversion—enter them into the new system or continue to use the old system for open rentals.

Training

The team estimates that only a few hours of training are required for clerks. With good design and help features, the system should be easy to use. The team recommends that GB not train the clerks on the system until a day or so before the clerks start to use the system. The system operator(s) can receive training from the vendor. The training plan in Table A.12 lists the GB responsibilities and dates for training.

Maintenance

The team understands that the vendor will maintain the production system code. GB will arrange for hardware maintenance. The planned responsibilities for maintenance appear in Table A.13.

Emergency Plan

While everyone hopes that emergencies will never happen, the team believes that GB should have a plan. In anticipation of a possible fire, tornado, major equipment failure,

TABLE A.13 Maintenance Plan for GB Video

Activity	Responsible Party
Hardware maintenance	Jim Croates
Program change control	AI Price
Log errors from users	Logged-on Web site
Review errors	AI Price
Track changes	AI Price
Make changes to code	Vendor
Changes in options and configuration	AI Price

TABLE A.14 GB Video Disaster Plan

Activity	Responsible Party
Identify secondary storage devices with critical data	IT Director
Set up call list for emergencies	IT Director
Ensure that all data and programs are backed up on a daily basis	Database Administrator
Set up a cold off-site center	Facilities Director
Set up password system for clerks	IT Director

or other disruption, GB needs to assign responsibilities for planning in advance to recover from any disruptive events. The assignments are shown in Table A.14.

APPENDIXES

Appendix A. Statement of Work

Appendix A contains the statement of work that the team prepared and discussed with the client. See Chapter 3 for the sample statement of work.

Appendix B. Forms for the Current Situation

Appendix B contains copies of currently used forms. The appendix contains the

- Member Data Card
- Invoice and Customer Receipt
- Video Rental Card

These forms appear in Chapter 7.

Appendix C. Current System Data and Process Models

Appendix C contains the graphical data and process models for the GB Video system. The models included in the appendix include:

- Exhibit 1. Context-Level DFD for GB Video Current Operation.
- Exhibit 2. First Explosion DFD for GB Video Current Operation.
- Exhibit 3. Metadata for the GB Video Current Operation DFD.
- Exhibit 4. Enterprise Data Model for GB Video.

The graphical models appear in Chapter 7.

Appendix D. Proposed Solution Data and Process Models

Appendix D contains the conceptual data and process models for the GB Video proposed system. The models included in the appendix include:

- Exhibit 1. Proposed System Narrative Specifications.
- Exhibit 2. Proposed System Data Flow Diagram.
- Exhibit 3. Proposed System Entity Relationship Diagram.
- Exhibit 4. Proposed System Metadata.

The graphical models appear in Chapter 8.

Appendix E. Request for Proposal

Appendix E contains the request for proposal. The request for proposal appears in Chapter 10.

Index

A

- Acceptance and credibility, of project team, 196
- Access control, for documents, 52
- Accuracy, of POC model, 431
- Action diagram, 405
- Action map, 405
- Active present-tense verbs, 97
- Activity/task
 - schedule table, 79–80
 - selection. *see* Generation of project plans
- Actors, use case diagrams, 179
- Adjourning stage, of team maturation, 38
- Agendas
 - project presentation slide, 100
 - for team, 51
 - for visits to clients, 218
- Alignment, of IT, with organization values.
 - see* Strategic alignment
- Alternative solutions, 208, 216, 300–329. *see also*
 - Evaluation, of alternative solutions
 - availability of, 308
 - client approval, 332
 - client preferences and, 299–300
 - current system and, 300
 - describing, 308–309
 - design options, 300–301
 - full description, requirements for, 309
 - functionality of, 301–302
 - infrastructure choices, 307
 - logic tests for, 457
 - outsourced system features, 365
 - performance evaluations, 307, 308
 - in project definition report, 208–209
 - recommended. *see* Recommendation report contents, 309
 - response times, 308
 - SDLC and, 18
 - slide, in project presentations, 100
 - solution classes, 6
 - sourcing options, 302–307
 - in-house system build, 303
 - outsourcing, 303–307
 - ASPs, 301, 306–307, 346
 - contract development, 306
 - package systems, 305–306
 - situations, 304
 - value tests at project completion, 457
 - zero-base design option, 301
- Analyst, 42
- Anchor, on presentation team, 99
- Anomalies, in logical data models, 145
- Appearance, of written reports, 98
- Application program interfaces (APIs), 349–350
- Application service provider (ASP), 301, 306–307, 346
- Approval, of work to date, 247
- Architecture, in alternative solutions report, 309
- Artifacts, POC models, 439
- Assignments, prototype-based POC design, 437–438
- Assistants, relationships with, 218
- Association, in class diagrams, 181
- Association model, for outsourcing, 341–342
- Associative entities, 125–126
 - data marts and, 408
- Atkinson, Anthony, 336
- Attributes
 - attribute metadata, 134–135
 - composite, 120
 - default, 182
 - derived, 121
 - in dimension table, 410–411
 - entity relationship model, 117
 - identification, simple ERD, 120–121
 - multivalued, 124
 - of object classes, 182
 - property strings, 182
 - slowly changing, 411
 - type attribute, 182
 - visibility attribute, 182
- Audit plan, post-implementation, 472
- Augustine, Norman, 111
- Automate (difference reduction) method, 261–262
- Automatic table/code generators, 67
- Availability, 308

B

Backups
 for copies of communication, 52
 for high risk solutions, 315

Baron, Robert A., 35, 38, 62

Basic function module (BFM), 392
 DFDs, 170

Behavior
 guidelines, 37
 professional, 215–216

Benefits. *see also* Evaluation, of alternative solutions
 estimating, 316–317
 evaluating, 315–319
 of structured system acquisition, 15

Bentley, Lonnie D., 10, 31, 153, 190, 295

Berra, Yogi, 220

Best practice, 28, 260–261

Best set, of modules, 391

Bids, outsourcing and RFPs, 356, 367

Binary relationship, 126–128

Boehm, Barry W., 111

Booch, Grady, 156, 190

Bottom-up POC design, 437

Box schema, 140

Brainstorming, 259–260

Break even analysis, 319–320

Brigham, Eugene, 336

Budget constraints, 208

Building a prototype. *see* Prototype-based POC design

Burd, Stephen D., 31

C

Calculation/Optimization, of proposed system,
 259, 262–263

Candidate primary key, 122

Cardinality, minimum, 128–129

CASE (computer-aided software engineering),
 26–27, 67

Central fact table, 408

Change
 analysis, retention and, 234–239
 inevitability of, 84
 in operations, 85–86
 requirements, controlling, 86–87
 resistance to, 85–86

Chen, P. P.-S., 114, 153

Class, entity relationship model, 117

Class diagrams, 26, 181–182, 288–291

Client(s), 6
 approval by, 332
 client/server systems, 68
 client/team contract approach, 69–70
 contact person, 218
 deliverables, statement of work and, 81
 directives, 313
 errors by, correcting, 219
 expectations, of the team, 52
 meetings with, project plans and, 76
 relationships with, managing, 53
 requirements and organization, 193–228
 strategic alignment. *see* Strategic alignment
 resources, statement of work and, 81
 roles, in system design, 379
 working with, 210–220
 behavior, professional, 215–216
 visiting with clients, 218–220

Closing the project, 474

COCOMO, 77

Codd, E. F., 136, 146, 153

Code generation
 CASE tools for, 67
 prototype-based POC design, 437, 438–440

Code modification, adding functionality, 350

Code of conduct, 42–43, 45

Cohesion, 376, 390

Column heading schema, 140

Commodity model, for outsourcing, 341–342

Commodity relationship, 350

Communication, 42, 91–104
 allowing client to finish sentences, 219
 electronic, 51–52
 presentations. *see* Presentations, project
 progress reports, 91–92
 within project teams, 36
 safe, alternative solutions and, 315
 working in a team and, 51–52
 written reports. *see* Written reports

Communicator, 42

Compaq Computer Co., 354

Comparison matrix, outsourcing, 341

Compatibility with physical infrastructure, 424,
 425, 430–431

Competency, of team, 247

Competition risk, 314

Competitive necessity, 313

- Completeness, 230
 - of final documentation, 461
 - narrative model of current situation, 239–240
- Completion of projects, 455–480
 - completeness of final documentation, 461
 - correctness of final documentation, 461
 - design specifications walk-through tests, 459
 - desk checks, 457
 - documentation clearance, 460–461
 - external walk-throughs, 459
 - format of final documentation, 460
 - implementation phase. *see* Implementation
 - internal walk-throughs, 458–459
 - live test data, 460
 - operational testing, 459–460
 - policies followed by final documentation, 461
 - post-implementation audit plan, 472
 - post-implementation tests, 460
 - readability of final documentation, 461
 - report and documentation tests, 457
 - solution logic tests, 457
 - solution value tests, 457
 - stub testing, 460
 - testing plans, 456–461
 - time to complete
 - increasing risk and, 315
 - as problem, 209
 - walk-through tests, 457–459
- Component model, of information systems, 9
- Component plan, prototype-based POC design, 438
- Composite attribute, 120
- Composite primary key, 123
- Compound process, DFDs, 170
- Comprehensive description, alternative solutions, 309
- Computer-assisted software engineering (CASE), 26–27, 67
- Conceptual data models (CDMs), 7, 18, 121, 131–133
 - proposed system, 258, 278–281
- Conceptual design cycle, spiral planning, 67
- Conceptual models, 11. *see also* Conceptual data models (CDMs)
- Conceptual specifications, 256–257
- Consistency rule, explosion DFDs, 174, 230
- Constantine, Larry, 156, 190, 376–377, 422
- Constraints on solutions
 - in project definition report, 207–209
 - project presentation slide, 100
- Constructed data, 460
- Consultants, corrective action and, 88
- Content emphasis, for prototypes, 435–436
- Content model, of information systems, 9–11
- Context-level DFD, 165–170, 240, 275
- Continuing operations, costs of, 318
- Contracts
 - contract development, 306
 - outsourcing and, 2, 350–351, 367
 - project team. *see* Project team contract
- Contributions of projects. *see also* Benefits; Evaluation, of alternative solutions
 - project definition report, 205
 - strategic alignment and, 200–201
 - strategic objectives and, 197–198
- Control, 84
 - execution and. *see* Project
- Control flow, POC model, 439
- Coordinator, 42
- Copies, to all members, 52
- Correcting client errors, 219
- Corrective action, 88
- Correctness, 230
 - of final documentation, 461
 - narrative model of current situation, 239–240
 - of project definition, 247
- Cost/benefit analyses, 311, 315–319
 - benefits, estimating, 316–317
 - continuing operations, costs of, 318
 - cost avoidance, 317
 - cost displacement, 317
 - cost reduction, 316–317
 - costs, identifying, 317–319
 - desirable features, 324, 343
 - evaluation summary table, 329
 - example of, 325–329
 - extreme risks, understanding, 315
 - facility costs, 318
 - feasibility analysis, 311–313
 - features analysis, 311, 324
 - implementation costs, 318
 - implied benefits method, 322–323
 - initial development costs, 318
 - intangible benefits, 317
 - internal rate of return (IRR), 315, 322
 - IT costs, 318
 - legal feasibility, 312
 - mandatory features, 324, 343
 - net present value (NPV), 320–322
 - “nice to have” features, 343
 - one-time costs, 318
 - ongoing costs, 318
 - operational feasibility, 312–313

- Cost/benefit analyses (*cont.*)
 - optional features, 324
 - outcome feasibility, 313
 - outsourcing evaluation metrics, 341
 - payback period, 315, 319–320
 - performance enhancement, 317
 - personnel costs, 318
 - project presentation summary slide, 100
 - return on investment (ROI), 322
 - revenue enhancement, 317
 - risk analysis, 313–315
 - risk estimates, 315
 - risk reduction, 314–315, 317
 - safe communication, 315
 - safe design practices, 314
 - schedule and cost feasibility, 312
 - security feasibility, 313
 - sponsor risk, 314
 - staff risk, 314
 - table for, 323–328
 - technical feasibility, 312
 - technology risk, 313–314
 - time-value-of-money methods, 320
 - total cost of ownership, 318
 - vendor risk, 313–314
 - Cost(s). *see also* Cost/benefit analyses
 - avoiding, 317
 - displacement of, 317
 - feasibility of solutions and, 312
 - identifying, 317–319
 - of outsourcing, 348–349
 - reducing, 316–317
 - of refinement/modification, 14
 - Cotterman, Howard, 111
 - Coupling, 390–391
 - Creating new objects (CURD), 177
 - Critical success measures, 92
 - CURD operations, 177, 182, 440
 - Current operations/systems/situation
 - alternative solutions and, 300
 - analysis, project definition and, 247–248
 - data models of. *see* Graphical data models
 - described, 233–234
 - DFDs for, 240
 - documenting, 222
 - EDMs for, 17–18
 - features of systems. *see* Features, proposed system
 - graphical data model of, 243–247
 - graphical process model of, 240–243
 - information collection on, 231–232
 - modification of, 265, 300
 - narrative model of. *see* Narrative model
 - project definition presentation, 247–248
 - replacing, systems development for, 6
 - retention and change analysis, 234–239
 - situation analysis goals, 230
- D**
- Data, 113
 - conversion, at implementation, 462
 - defined, 8
 - features, 344
 - in narrative model of current situation, 233
 - outputting, pseudocode and, 396
 - system design and, 375
 - transforming, pseudocode and, 393
 - Databases, 23
 - Data control language, 146
 - Data coupling, 390–391
 - Data CRUD operations, 177, 182
 - POC models, 440
 - Data definition language (DDL), 145–146
 - Data-driven development, 23–24
 - Data-driven systems, 387
 - Data flow, 157, 158
 - diagrams. *see* Data flow diagrams (DFDs)
 - dialog-driven systems, 403
 - in narrative model of current situation, 233
 - Data flow diagrams (DFDs), 10, 24–25, 157–174.
 - see also* Object models
 - basic function module (BFM), 170
 - building, 159–162
 - compound process, 170
 - consistency rule, 174
 - context-level, 275
 - for current operations, 240
 - data flow symbol, 158
 - for a data mart, 417
 - data store completeness rule, 163
 - data store sufficiency rule, 164
 - data store symbol, 158
 - decomposition rule, 174
 - elementary process (EP), 170
 - ERD/DFD integration, 274
 - external symbol, 159
 - as graphical process models, 239–240
 - hierarchical. *see* Hierarchical data flow diagrams

- labeling rules, 163–164, 174
- object models. *see* Object models
- object-oriented design (OOD) versus, 177
- physical, 389
- primitive process, 170
- process completeness rule, 163
- process data sufficiency rule, 164
- process dominance rule, 162
- process symbol, 159
- rules for, 162–165
- split data flow, 164–165
- sufficiency rules, 164
- symbols used in, 158–159
- Data input
 - form, 115
 - pseudocode and, 393
- Data items, 382
 - metadata for, 383
- Data manipulation language (DML), 146
- Data mart, 146, 408, 412–417
- Data mining, 147
- Data models, 17, 113–154, 243
 - conceptual (CDMs). *see* Conceptual data models (CDMs)
 - of current situation, 243–247
 - EDMs. *see* Enterprise data models (EDMs)
 - entity. *see* Entity relationship model (ER model)
 - ERDs. *see* Entity relationship diagrams (ERDs)
 - graphical. *see* Graphical data models
 - logical. *see* Logical data models
 - metadata, 133–136
 - POC models, 439
- Data output form, 115
- Data owners, roles in system design, 379
- Data schema, system design and, 386
- Data section, of content model, 10
- Data specifications, in proposed system, 267–268
- Data stores, 241
 - completeness rule, DFDs, 163
 - dialog-driven systems design, 403
 - operations on, pseudocode and, 393–394
 - sufficiency rules, DFDs, 164
 - symbol, DFDs, 158
- Data structure
 - alternative solutions report, 309
 - systems design and. *see* System design
 - traditional model, 115
- Data types, 383
- Data warehouse, 146–147, 408–418
 - dimensional models, 408–411
 - ETL process, 411–418
 - metadata for, 411, 412–417
- Date dimension, fact tables, 410
- DEC, 354
- Decision making, strategic alignment and, 196
- Decomposition rule, explosion DFDs, 174
- Default attribute, of object classes, 182
- Degrees, of a relationship, 126–128
- Deleting objects (CURD), 177
- Deletion anomaly, 145
- Deliverables, 7
- Delivery, of project presentations, 102–103
- Delivery record, vendors, 352, 353
- DeMarco, Tom, 376–377, 422
- Departures (exemptions) from plan, 91–92
- Derived attributes, 121
- Design
 - approaches, proposed system, 264–266
 - option, for alternative solutions, 300–301
 - parameter evaluation, POC model, 430–431
 - safe practices, 314
 - specifications
 - coding and, POC models, 438–440
 - walk-through at project completion, 459
- Designer, 42
- Desirable features, 324, 343
- Desk checks, at project completion, 457
- Development activities, 21–27
 - CASE tool-driven, 26–27
 - data-driven, 23–24
 - DFDs. *see* Data flow diagrams (DFDs)
 - ERDs. *see* Entity relationship diagrams (ERDs)
 - event-driven, 25–26
 - field project challenges, 27–28
 - OOD. *see* Object-oriented design (OOD)
 - output-driven, 22–23
 - process-driven, 24–25
 - prototyping. *see* Prototyping
 - structuring of, 27
 - technology-driven, 21–22
- Diagram symbols. *see* Symbols
- Dialog-driven systems, 387, 399–407
 - data flows, 403
 - data stores, 403
 - externals, 403
 - links, 402
 - menus, 402
 - page action maps, 399, 404–407

Dialog-driven systems (*cont.*)
 page navigation maps, 399, 402–404, 407
 page representations, 402
 processes, 403

Difference reduction methods, 259, 261–262

Dimensional models, 146–147
 data warehouse design, 408–411

Dimension table, 410–411

Direct implementation, 463–464, 465

Disaster plans, 471–472

Discover (difference reduction) method, 261–262

Disjoint supertype links, 130

Dittman, Kevin C., 10, 31, 153, 190, 295

Documentation
 clearance of, at project completion, 460–461
 at implementation phase, 469–471
 for information collection, 222
 SDLC, 18

Due date (end date), 77, 78
 extensions of due date, 78, 88

Dysfunctional teams. *see* Nonperforming/
 dysfunctional members

E

Economic value analysis, 363–364

Editor, 42

Education. *see also* Training
 regarding client problems, 217

Effective coding, POC models, 440

Effectiveness
 improving, 88
 of project teams, 36, 47–48

Efficiency, of outsourced products, 345

Electronic communications, 51–52

Elementary modules, 392

Elementary process (EP), DFDs, 170

E-mail, 51–52
 in place of meetings, 218
 to schedule client visits, 217–218

Emerging technologies, project risk and, 315

Emery, Douglas, 336

Encapsulated procedures/operations, 178

End of visits, managing, 220

Enterprise data models (EDMs), 133–136
 attribute metadata, 134–135
 correspondence with relational models, 137
 for current operations, 17–18
 data models, 133–136
 entity metadata, 134
 graphical data model of current operation, 246
 metadata, 133–136
 relationships, 135
 rules for, 136
 in use case diagrams, 180

Enterprise resource planning (ERP), 342, 381

Entities, 115–117. *see also* Entity relationship
 diagrams (ERDs); Entity relationship
 model (ER model)
 associative, 125–126
 entity identification, simple ERD, 119
 weak, 124–125

Entity metadata, 134

Entity relationship diagrams (ERDs), 23–24, 114,
 117–131, 344. *see also* Entity relationship
 model (ER model)
 additional constructs for, 124–130
 associative entities, 125–126
 converting to relational schema, 141–142
 degree of relationship, 126–128
 DFD integration with, for MDFDs, 274
 as graphical data models, 239–240
 multivalued attributes, 124
 naming rules, 123–124
 rules/guidelines for, 122–124
 simple, 118–121, 130–131
 subtypes, 129–130
 supertypes, 129–130
 symbols for, 117–118
 weak entities, 124–125

Entity relationship model (ER model). *see also*
 Entity relationship diagrams (ERDs)
 attribute identification step, 120–121
 conceptual data models (CDMs), 121
 entity identification step, 119
 logical data models, 121
 maximum cardinalities, 120
 minimum cardinality, 128–129
 model components, 115–117
 physical data models, 121
 relational model terms, compared, 137
 relationship identification step, 120

Environmental issues, 20

Evaluation, of alternative solutions, 309–329.
see also Alternative solutions
 backup for high risk solutions, 315
 break even analysis, 319–320
 client directive and, 313
 comparing alternatives, 324

- competition risk, 314
 - competitive necessity and, 313
 - cost/benefit analyses. *see* Cost/benefit analyses
 - Evaluation metrics. *see* Evaluation, of alternative solutions
 - Event-driven development, 25–26
 - Events, in narrative model, 233
 - Evolution, of project teams, 37–38
 - Evolutionary prototypes, 72, 428, 434–435
 - Execution and control, for projects. *see* Project
 - Executive summary, written reports, 96
 - Experience, lack of, project risk and, 315
 - Experienced-based methods, proposed system, 259–260
 - Explosion DFDs, 165, 170–174
 - consistency rule, 174, 230
 - context-level, 172–174
 - decomposition rule, 174
 - first-level explosion, 170–172, 240–242, 344
 - labeling of, 174
 - rules, additional, 174
 - Explosion MDFDs, 275–278
 - Extensions, of due dates, 88
 - Externals, dialog-driven systems design, 403
 - External symbol, DFDs, 159
 - External walk-throughs, project completion, 459
 - Extraction-transform-load process, 411–418
- F
- Face-to-face discussions, 51
 - Facilitation vs. domination, 37
 - Facility costs, 318
 - Fact tables, 408, 410
 - Familiarity, lack of, project risk and, 315
 - Feasibility
 - analysis, 311–313
 - infeasibility of statements of work, in the future, 84
 - legal, 312
 - operational, 312–313
 - POC model, 424–425, 429–430
 - outcome, 313
 - schedule and cost, 312
 - security, 313
 - technical, 312
 - Features, proposed system, 198, 201, 255–296, 311, 324, 363
 - analysis matrix, 341, 363
 - of an object class, 182
 - best practice, 260–261
 - brainstorming for, 259–260
 - calculation/optimization methods, 259, 262–263
 - concepts for, 258–266
 - conceptual data models (CDM) for, 258, 278–281
 - conceptual specifications, 256–257
 - cost/benefit analysis of. *see* Cost/benefit analyses
 - data, 344
 - design approaches, 264–266
 - desirable, 324, 343
 - difference reduction methods, 259, 261–262
 - evaluation of, 2
 - experienced-based methods and, 259–260
 - functional product, 344
 - generation of project plans, 75
 - goals and outcomes, 257–258
 - graphical data models, 258
 - graphical data specifications, 278–281
 - graphical object models, 258
 - graphical process models, 258
 - graphical process specifications. *see* Graphical process specifications
 - heuristic methods and, 259, 260–261
 - impact of, 206–207
 - list, 363
 - mandatory, 324, 343
 - matrix, 341, 363
 - MDFDs. *see* Modified data flow diagrams (MDFDs)
 - metadata and, 281–282
 - modification of current system, 265
 - narrative specifications. *see* Narrative model
 - narrative statements, 258
 - new design vs. modifications, 265–266
 - “nice to have,” 343
 - object-oriented design (OOD) and, 258, 282–291
 - class diagrams, 288–291
 - use case diagrams, 287–288
 - optional, 324
 - organizational models and, 263–264
 - outsourcing and. *see* Outsourcing
 - problem-solving methods. *see* Problem-solving methods
 - process, 344
 - of products. *see* Products
 - in project definition report, 205–207
 - ranking methods, 363–364
 - ratings assigned to, 364, 365–367
 - review of, 89
 - SDLC, 18

Features, proposed system (*cont.*)

- security, 378
 - slide, in project presentations, 100
 - specifications defined, 258–259
 - Value Chain Model, 263–264
 - vendor, 352–353
 - weighted features analysis, 364, 366
- Feedback
- at project presentations, 247
 - at visits with clients, 220
- Field project challenges, 27–28
- File metadata, 383
- Final deliverables, 19
- Final design review, 89
- Final presentation, 103–104
- Financial sponsorship, 196
- Finnerty, John D., 336
- First-level explosion
- DFDs, 170–172, 240–242, 344
 - MDFDs, 275–278
- First normal form, logical data models, 144
- Fixed period software license, 347–348
- Fixed price, outsourcing and, 348
- Flaming, 52
- Flexibility, of project team members, 37
- Flexible project planning, 67–69
- Flow of control, pseudocode and, 395–396
- Focus groups, 222
- Follow-on training, 466, 467–468
- Foreign key, 137, 139
- Format
- of final documentation, 460
 - for narrative model, 266–269
- Forming stage, of team maturation, 38
- Forsberg, Kevin, 64, 111
- Fourth-generation languages (4GLs), 67
- Fowler, Martin, 10, 31, 157, 190
- Fragmentation, 55–56
- Frequency, of reports, 92
- Functional dependency, 144
- Functionality. *see also* Functions
- alternative solutions and, 301–302
 - POC model, 424, 429–430
 - of product features, 344
 - as a variable, 301
- Function hierarchy diagrams (FHDs), 10.
see also Process(es)
- Functions, 206. *see also* Functionality
- alternative solutions and, 301
 - constraining solutions, 208

G

- Gane, Chris, 156, 190
- Gantt charts, 16
- project plan generation and, 78–79
- Gapenski, Louis, 336
- General Electric, 354
- Generated surrogate key, 409
- Generation of project plans, 73–80
- activity schedule table, 79–80
 - activity/task selection, 74–76
 - project and team organization, 75
 - project definition, 75
 - proposed system, 75
 - system delivery, 75
 - activity/task times and sequence, 76–77
 - client meetings, 76
 - constructing the schedule, 77–80
 - Gantt chart schedule, 78–79
 - person-hours of work required, 77
 - slack, 78
 - slippage, 78
 - start date and end date (due date), 77, 78
 - extensions of due date, 88
- George, Joey F., 10, 31, 153, 190, 295
- Goals, 197
- current situation analysis, 230
 - in project definition report, 204–205
 - of project teams, 35–36
 - proposed system specifications, 257–258
 - strategic alignment and, 197, 199–200
- Good writing, for written reports, 96–97
- Graphical data models, 239–240
- of current operation, 243–247
 - object models, 258
 - process models, 239–243, 258
 - metadata, 243, 244–245
- Graphical data specifications, 258, 267, 278–281
- Graphical process specifications, 269–278
- context-level DFDs, 275
 - MDFDs. *see* Modified data flow diagrams (MDFDs)
- Graphic user interfaces (GUI), 67
- Greenberg, Jerald, 35, 38, 62
- Group interviews, 221–222

H

- Handouts, 51
- Hardware specifications, 378
- Headings and fonts, written reports, 95–96

Headship, for project teams, 43
 Help desk, 467
 Heuristic methods, 259, 260–261
 Hewlett-Packard, 10
 Hierarchical data flow diagrams, 165–174
 added explosion levels, 172–174
 context-level, 165–170, 240
 first-level explosion, 170–172, 240–242, 344
 rules for, 169–170, 174
 written narrative prepared from, 168
 Hijacked teams, 56
 Historical role of information, 4–5
 Hoffer, Jeffrey A., 10, 31, 114, 146, 153, 156, 190, 295
 Host, on presentation team, 99

I

IBM, 70
 Immediacy, MDFDs and, 274
 Impact of a feature, 206–207
 Implementation, 461–474
 closing the project, 474
 costs of, 318
 data conversion, 462
 direct implementation, 463–464, 465
 disaster plans, 471–472
 documentation, 469–471
 GB Video example, 472–474
 help desk, 467
 issues slide, project presentations, 101
 maintenance plan, 469
 parallel, 462–463, 464, 465
 phased, 463, 464–465
 pilot, 463, 464–465
 plans/strategies for, 461–469
 post-implementation audit plan, 472
 SDLC and, 19
 sequential approaches, 464–465
 system controls, 471
 training, 465–469
 Implied benefits, 322–323
 Incentive pricing, outsourcing and, 348
 Individual needs, team motivation and, 48–50
 Inexpensive trial solutions, 376
 Infeasibility, of statements of work, 84
 Information
 confidentiality of, 219
 historical role of, 4–5
 Information age, 4
 Information collection, 195, 220–223
 on current situation, 231–232
 documents, 222
 focus groups, 222
 group interviews, 221–222
 interviews, 221
 observation, 222–223
 sampling, 223
 surveys, 223
 Information processing, 4–5
 Information system life cycle, 12–14
 Information systems. *see also* Information system solutions
 benefits of, determining, 14
 development. *see* Development activities
 life cycle of, 12–14
 proposed. *see* Features, proposed system
 roles for, 12
 solutions. *see* Information system solutions
 use and refinement, 14
 Information Systems Group (IS), 6
 Information system solutions, 5–15. *see also*
 Alternative solutions; Information systems
 component model, 9
 concepts and models for, 8–11
 content model, 9–11
 design of, 5
 models. *see* Models
 system solution activities, 6–8
 technology level of models, 11
 Information technology (IT), 4, 6, 12
 activities vs. organizational values. *see* Strategic alignment
 cost/benefit analysis. *see* Cost/benefit analyses
 outsourcing of, 72–73, 303–307
 typical project, 5
 Inform (difference reduction) method, 261–262
 Infrastructure options, 301, 307
 In-house system builds, 303
 Initial analysis, spiral planning model, 67
 Initial development costs, 318
 Initial prototype-based POC design, 436–437
 Initial role assignments, 41–42
 Initial training, 466–467
 Inmon, W. H., 408, 422
 Input/process/output charts, 157, 174–175
 Input(s)
 of data, 115, 393
 input/process/output charts, 157, 174–175
 to outsourcing task, 340

Insertion anomaly, 145
 Instance, entity relationship model, 117
 Institution-specific applications, 303
 Intangible benefits, 317
 Integration, of theory and applications, 11
 Interactions/critiques from client, encouraging, 220
 Internal rate of return (IRR), 315, 322
 Internal walk-through, 458–459
 Interoperability, 307, 346
 Interviews, 221
 Introductions
 in proposed system narratives, 267
 in written reports, 96
 IPO (input/process/output) charts, 157, 174–175
 IT. *see* Information technology (IT)

J

Jacobson, Ivar, 156, 190
 Joint team approach, project planning, 70
 “Just-do-it” system acquisition model, 15, 16

K

Kendall, Julie E., 228
 Kendall, Kenneth E., 228
 Kimball, Ralph, 408, 422

L

Labeling
 of DFDs, 163–164, 174
 of MDFDs, 274
 Laws and regulations
 legal feasibility, 312
 outsourcing and, 349
 Leadership, 43, 50–51
 Legal feasibility, 312
 Lifetime software license, 347
 Links, 402
 Listening skills, 220
 Live test data, 460
 Logic, POC models, 440
 Logical data models, 121, 136–147
 anomalies in, 145
 data control language, 146
 data definition language (DDL), 145–146
 data manipulation language (DML), 146

 dimensional models, 146–147
 normalization of, 144–145
 relational model, 136–139
 basic concepts, 137–138
 correspondence with ER models, 137
 foreign key, 137, 139
 relational schema. *see* Relational schema
 rules for, 138–139
 table, 137
 unique key, 137
 standard query language (SQL), 145–146
 Logical design cycle, spiral planning model, 67
 Logical models, 11. *see also* Logical data models
 Logic and functionality section, alternative
 solutions report, 309

M

Mainframes, 68
 Maintainability
 of outsourced products, 346–347
 system design and, 376
 Maintainers, role in system design, 379
 Maintenance plan, 469
 Management of project teams. *see* Project team
 Manager deliverables, statement of work and, 81
 Manager relations, working in a team, 52–53
 Mandatory items
 features, 324, 343
 processes, 268
 relationship, 128
 Many-to-many relationships, 120
 unary, 143
 March, James G., 299, 311, 336
 Marginal net benefits, 13, 14
 Maturation, of project teams, 37–38
 Maximum cardinality, 120
 McClelland, David C., 48, 50, 62
 McFadden, F. R., 10, 31, 153, 295
 Meaning checking, 97–98
 Meetings, code of conduct, 42
 Members of teams. *see* Project team
 Menus, dialog-driven systems design, 402
 Messages
 coupling, 390–391
 object interactions and, 178
 Metadata, 133–136, 241
 attribute metadata, 134–135
 data items, 383

- data models, 133–136
 - data structure, 382–386, 389, 396–399
 - data structure and system design, 382–386, 389
 - for data warehouse, 411, 412–417
 - entity metadata, 134
 - file metadata, 383
 - for module logic, 396–399
 - page navigation/action maps, 407
 - process model, 243, 244–245, 389
 - proposed system example, 283–287
 - proposed system features/specifications, 281–282
 - relationships, 135
 - specifications, proposed system, 281–282
 - table, 383
 - in use case diagrams, 180
 - Microsoft Project, 90–91
 - Minimum cardinality, 128–129
 - Mission statement, 198
 - Models, 8–11
 - commodity, for outsourcing, 341–342
 - component-based, 9
 - conceptual, 11. *see also* Conceptual data models (CDMs)
 - content-based, 9–11
 - content of information systems, 9–11
 - data. *see* Data models
 - dimensional. *see* Dimensional models
 - enterprise data. *see* Enterprise data models (EDMs)
 - entity relationship (ER model). *see* Entity relationship model (ER model)
 - graphical data. *see* Graphical data models
 - logical, 11. *see also* Logical data models
 - metadata and. *see* Metadata
 - narrative model. *see* Narrative model
 - object. *see* Object models
 - organizational, proposed features and, 263–264
 - of outsourcing, 341–342
 - physical, 11, 121
 - physical data, 121
 - POC. *see* Proof of concept (POC) models
 - relational, ER model compared, 137
 - technology level of, 11
 - Modification anomaly, 145
 - Modification of current system, proposed system and, 265
 - Modified data flow diagrams (MDFDs), 258, 269–275
 - DFD/ERG integration, 274
 - explosions
 - additional, 278
 - first, 275–278
 - first explosion MDFD, 275–278
 - guidelines for creating, 274–275
 - labeling, 274
 - process triggers, 274
 - time and immediacy, 274
 - triggers, 269–275
 - Module design, 387, 390–399
 - basic functional modules, 392
 - cohesion and, 390
 - coupling, 376, 390–391
 - data coupling, 390–391
 - elementary modules, 392
 - message coupling, 390–391
 - metadata for module logic, 396–399
 - pseudocode. *see* Pseudocode
 - satisficing and, 391
 - specifications for, 391–392
 - TIPOT charts, 399, 400–401
 - Monitor progress, 87–88
 - Mooz, Hal, 111
 - Morris, Chuck (IBM), 70
 - Motivating project teams, 47–51
 - individual needs and, 48–50
 - leadership, motivational perspective, 50–51
 - need for achievement (nAch), 48–49
 - need for affiliation (nAff), 49
 - need for power or influence (nPow), 49–50
 - Multiple representations, 239–240
 - Multiplicity attributes, 182
 - Multivalued attributes, 124
 - Mutual accountability, 36
- N**
- Naming rules
 - ERD, 123–124
 - POC models, 439–440
 - Narrative model
 - current situation, 232–240
 - correct/complete representations, 239–240
 - data content, 233
 - data flows and processes, 233
 - description of, 233–234
 - events, 233
 - organizational infrastructure, 234
 - physical infrastructure, 234

- Narrative model (*cont.*)
 - current situation (*cont.*)
 - problem analysis, 234
 - retention and change analysis, 234–239
 - proposed system
 - introduction, 267
 - data specifications, 267–268
 - process specifications, 268
 - organizational specifications, 269
 - example, 270–273
 - format for, 266–269
 - mandatory processes, 268
 - optional processes, 268
 - Need for achievement (nAch), 48–49
 - Need for affiliation (nAff), 49
 - Need for power or influence (nPow), 49–50
 - Negative net benefit, 14
 - Net benefit, 323
 - Net marginal value, 13, 14
 - Net present value (NPV), 14, 320–322
 - Network diagram, 91
 - Network impacts, system design, 378
 - New design, vs. modifications, 265–266
 - Nonperforming/dysfunctional members, 53–57
 - appropriate handling of, 54
 - described, 53
 - fragmentation of teams, 55–56
 - hijacked teams, 56
 - removal of a member, 54–55
 - resignation of a team member, 55
 - the secretive genius, 56–57
 - Normalization, of logical data models, 144–145
 - Norming stage, of team maturation, 38
 - Note taking
 - on presentation team, 99
 - in visits with clients and, 220
- O**
- Object class, 177, 182
 - in class diagrams, 181
 - CURD operations and, 182
 - Objectives, 197
 - project definition report and, 204–205
 - strategic, 197–200
 - strategic alignment and, 197, 199–200
 - The Object Management Group, 178
 - Object models, 155–157, 176–184. *see also* Object-oriented design (OOD)
 - class diagrams, 181–182
 - objects defined, 156, 177
 - sequence diagrams, 183–184
 - Unified Modeling Language (UML), 156
 - use case diagrams, 178–180
 - Object-oriented design (OOD), 26. *see also* Object models
 - advantages of, 184
 - specifications, proposed. *see* Features, proposed system
 - Object plan, prototype-based POC design, 438
 - Objects, 156, 177
 - Observation, for information collection, 222–223
 - One-time costs, 318
 - One-to-many relationships, 120
 - unary, 143
 - One-to-one relationships, 120
 - unary, 143
 - Ongoing costs, 318. *see also* Cost/benefit analyses
 - Operational feasibility
 - alternative solutions, 312–313
 - POC model, 424–425, 429–430
 - Operational POC model, 424
 - Operational product features. *see* Products
 - Operational testing, at project completion, 459–460
 - Operations, 177
 - in class diagrams, 181
 - data stores and pseudocode, 393–394
 - project team code of conduct, 42
 - Operators, role in system design, 379
 - Optimization, for problem solving, 263
 - Optional features, 324
 - Optionality, 128–129
 - Optional processes, 268
 - Optional relationship, 128
 - Oracle, 10, 22, 26, 378, 379, 381, 427
 - Organization, 6
 - infrastructure. *see* Organizational infrastructure
 - organizational case, 196–198
 - project definition report, 204–205
 - strategic alignment and, 197. *see also* Strategic alignment
 - Organizational case, 196–198
 - Organizational culture, 199
 - Organizational fit, of outsourced product, 351

- Organizational infrastructure, 10–11
 - example, 380–381
 - narrative model of current situation, 234
 - system design and, 375, 378–379
 - Organizational models, proposed features and, 263–264
 - Organizational specifications, in system narrative, 269
 - Organization constraints, on solutions, 208
 - Outcomes
 - feasibility of alternatives, 313
 - outsourcing and, 367
 - of proposed systems, 257–258
 - Output
 - of data, pseudocode and, 396
 - data output form, 115
 - input/process/output charts, 157, 174–175
 - output-driven development, 22–23
 - Outsourcing, 72–73, 303–307, 339–369
 - association model, 341–342
 - candidate solutions, identifying, 365
 - commodity model, 341–342
 - comparison matrix, 341
 - contracts for products/services, 367
 - economic value analysis, 363–364
 - evaluation metric, 341
 - features analysis, 2, 341, 363
 - weighted, 364, 366
 - inputs to outsourcing task, 340
 - models of, 341–342
 - outcomes, 367
 - process of, 341–343
 - product features and. *see* Products
 - ranking methods, 363–364
 - rating points, 364
 - ratings, assigning, 365–367
 - request for quote (RFQ), 341
 - requirements, refining, 341, 343–351. *see also* Products
 - RFP. *see* Request for proposal (RFP)
 - vendor roles. *see* Vendors
 - weighted features analysis, 364, 366
 - Overlap supertype links, 130
 - Ownership of documents, 52
- P**
- Package systems, 73, 305–306
 - package POC models, 426–428
 - Page action maps, 399, 404–407
 - Page navigation maps, 399, 402–404, 407
 - Page representations, dialog-driven design, 402
 - Parallel implementation, 462–463, 464, 465
 - Partial specialization, supertypes, 130
 - Payback period, 14, 319–320
 - shorter, for riskier projects, 315
 - Payments, for supportive behavior, 50
 - Peer evaluations, 57–59
 - PeopleSoft, 381, 427
 - Performance, 206
 - alternative solutions and, 301
 - enhancing, 317
 - infrastructure choices and, 307, 308
 - measuring, 198, 201
 - of outsourced products, 345
 - performance-oriented design, 198
 - Performing stage, of team maturation, 38
 - Person-hours required, generation of project plans, 77
 - Personnel costs, 318
 - PERT/CPM charts, 16, 77, 91
 - Phased implementation, 463, 464–465
 - Physical data flow diagrams, 389
 - Physical data models, 11, 121
 - Physical design cycle, spiral planning, 67
 - Physical infrastructure
 - of an information system, 10
 - as constraint on solutions, 208
 - example, 380
 - narrative model of current situation, 234
 - system design and, 375, 377–378
 - Physical models, 11, 121
 - Pilot implementation, 463, 464–465
 - Ping-pong paragraphs, avoiding, 97
 - Piracy of software, 350
 - Planning and management, 15. *see also* Project Management (PM); Project planning
 - POC models. *see* Proof of concept (POC) models
 - Policies, final documentation following, 461
 - Popkin, 26
 - Porter, Michael E., 295
 - Post, Gerald V., 31, 114, 146, 153, 295
 - Post-implementation audit, 298, 472
 - Post-implementation reviews, 89–90
 - Post-implementation tests, 460
 - Pratt, Philip J., 146, 153
 - Pre-implementation reviews, 89
 - Prescott, M. B., 10, 31, 153, 295

- Presentations, project, 98–104
 - delivery of, 102–103
 - final presentation, 103–104
 - guidelines for, 98–99
 - presenter, on presentation team, 99
 - project definition presentation, 247–248
 - rehearsals, 101
 - roles of team members, 99
 - setup tips, 102
 - visual aids, 99–101
- Presenter, on presentation team, 99
- Present value, 320. *see also* Net present value (NPV)
- Primary key, 122
 - data marts and, 409
 - entity relationship model, 117
- Primitive process, DFDs, 170
- Problem analysis, narrative model, 234
- Problem-solving methods, 259–262
 - automate (difference reduction), 261–262
 - best practice, 260–261
 - brainstorming, 259–260
 - calculation and optimization, 259, 262–263
 - difference reduction (discover), 259, 261–262
 - experienced-based, 259–260
 - heuristics, 259, 260–261
 - inform (difference reduction), 261–262
 - optimization, 263
 - process specifications identified using, 268
 - transform (difference reduction), 261–262
 - trial and error, 259, 260
- Procedures
 - encapsulated, 178
 - performing, pseudocode and, 394
- Process(es), 8
 - completeness rule, DFDs, 163
 - data sufficiency rules, DFDs, 164
 - design of, 376, 387–389
 - dialog-driven systems design and, 403
 - dominance rule, DFDs, 162
 - features, 344
 - hierarchy charts, 18–19, 24, 157, 175–176
 - modeling, 17, 155–176, 239–240
 - DFDs. *see* Data flow diagrams (DFDs)
 - input/process/output charts, 157, 174–175
 - metadata, 243, 244–245, 389
 - process models defined, 156
 - process-driven development, 24–25
 - section, of content model, 10
 - specifications, proposed system, 268
 - symbol, DFDs, 159
 - triggers, MDFDs and, 274
- Procurement options, 208
- Production programs, from proof of concept, 428–429
- Products
 - data features, 344
 - flexibility of, outsourcing and, 349–350
 - functional, 344
 - operational features, 344–351
 - contract compliance, 350–351
 - costs and prices, 348–349
 - efficiency, 345
 - fixed period software license, 347–348
 - interoperability, 346
 - laws and regulations, 349
 - lifetime software license, 347
 - maintainability, 346–347
 - organizational fit, 351
 - performance, 345
 - product flexibility, 349–350
 - security, 346
 - software licenses, 347–348
 - time-bomb components, 348
 - usability, 345
 - process features, 344
 - selling, 12
- Professional behavior, 215–216
- Professional integrity, 216
- Program structure charts (PSCs), 387–389
- Progress
 - informing team manager of, 53
 - versus plan, monitoring, 87–88
 - reports, 91–92, 93
- Project, 1, 5. *see also* Project definition; Project planning
 - approach, historical role of information, 4–5
 - communication. *see* Communication
 - description, statement of work and, 80
 - director of, 47
 - execution and control, 84–90
 - change, inevitability of, 84
 - changes in operations, controlling, 85–86
 - corrective action, 88
 - monitoring progress against the plan, 87–88
 - project execution explained, 85
 - requirements, controlling changes in, 86–87
 - resistance to change, 85–86
 - review points, 89–90
 - organization, 75

- project teams. *see* Project team
- schedules, statement of work and, 81
- successful. *see* Successful projects
- Project definition, 191–252. *see also* Project definition report
 - completion of, 249
 - explained, 191, 194
 - generation of project plans, 75
 - information collection. *see* Information collection
 - materials, examples, 210, 211–215
 - presentation, current situation, 247–248
 - questions to be answered, 194
 - review, 89
 - SDLC, 17–18
 - strategic alignment. *see* Strategic alignment
 - working with the client. *see* Client(s)
- Project definition report, 202–210
 - constraints on solutions, 207–209
 - the organization section, 204
 - the organization's goals/objectives section, 204–205
 - procurement options, 208
 - project contribution, 205
 - project definition materials, 210, 211–215
 - project statement, 203–204
 - project success criteria, 205
 - project success index, 210
 - proposed system features, 205–207
 - scope of the project, 209–210
 - solution options, 208–209
 - strategic alignment and, 204
- Project management (PM), 8, 15–21, 63–112
 - communication. *see* Communication
 - execution and control. *see* Project
 - project planning. *see* Project planning
 - project teams. *see* Project team
 - SOW. *see* Statement of work (SOW)
 - structure and flexibility, balancing, 20–21
 - SDLC. *see* Systems development life cycle (SDLC)
 - tools, 90–91
- Project Management Institute, 15
- Project manager, 47
- Project organization, SDLC, 16–17
- Project planning, 64–80
 - CASE tools, 67
 - client/team contract approach, 69–70
 - flexible project planning, 67–69
 - generating the plan. *see* Generation of project plans
 - joint team approach, 70
 - outsourcing of IT, 72–73, 303–307
 - packaged applications, purchasing, 73
 - planning mechanisms, 69–73
 - proof of concept versions of systems, 72
 - prototype-based plans, 70–72
 - rapid development planning model, 20–21, 68–73
 - SDLC for, 65, 67
 - spiral model, 66–67
 - team member turnover, 69
- Project statement
 - in project definition report, 203–204
 - slide, in project presentations, 100
- Project success. *see* Successful projects
- Project team, 6, 33–62
 - behavior guidelines, 37
 - code of conduct, 42–43, 45
 - communication within, 36
 - contract. *see* Project team contract
 - contributions of team members, 34
 - defined, 33
 - effective, 36–38, 47–48
 - evolution of, 37–38
 - facilitation, not domination, 37
 - flexibility of members, 37
 - goals of, 35–36
 - initial impressions of, 34
 - management of, 36, 43–51
 - headship, 43
 - leadership, 43
 - project director, 47
 - project manager, 47
 - team lead, 47
 - maturing of, 34, 37–38
 - members of, 34, 35, 37, 47
 - best skilled person, 88
 - motivating. *see* Motivating project teams
 - nonperforming. *see* Nonperforming/dysfunctional members
 - organization of, SDLC and, 16–17
 - roles of, 99
 - turnover of, 69
 - mutual accountability, 36
 - obligations of members, meeting, 37
 - peer evaluations, 57–59
 - punctuated-equilibrium maturation, 38
 - skill sets required, 36, 39–41, 46
 - speaking up, 37
 - start up, 36–37
 - successful, 36–38, 47–48
 - theory and principles, 35–36
 - working in a team, 51–53

- Project team contract, 39–43
 - analyst, 42
 - code of conduct, 42–43, 45
 - communicator, 42
 - coordinator, 42
 - designer, 42
 - editor, 42
 - role assignments, 41–42
 - sample of, 44
 - skills inventory, 36, 39–41, 46
 - standards manager, 42
 - team self-management and, 39
 - writer, 42
 - Proof of concept (POC) models, 423–454
 - accuracy of, 431
 - compatibility with physical infrastructure, 424, 425, 430–431
 - configuration of demo setups, 427
 - cost of, 427
 - decision variables, 427
 - design parameter evaluation, 430–431
 - effort to install and initialize the demo, 427
 - evolutionary prototypes, 428
 - functionality, 424, 429–430
 - infrastructure to run model, 427
 - operational feasibility, 424–425, 429–430
 - operational POC model, 424
 - package POC models, 426–428
 - production programs, 428–429
 - prototype-based design. *see* Prototype-based POC design
 - purposes of, 424–425
 - schedule for obtaining, 427
 - simulated POC models, 426
 - slides for project presentations, 101
 - static POC model, 424
 - system design parameters confirmation, 424, 425
 - system specification, refining, 424, 425
 - throwaway prototypes, 71–72, 428, 434
 - types of POC models, 425–429
 - usability, 424, 429–431
 - validity of, 431
 - Properties, of object classes, 182
 - Property string attribute, 182
 - Proposed system features. *see* Features, proposed system
 - Prototype-based POC design, 425, 428–429, 431–452
 - building a prototype, 434–440
 - artifacts, 439
 - bottom-up plan, 437
 - code, generation of, 437, 438–440
 - component plan, 438
 - control flow, 439
 - data CRUD operations, 440
 - data model, 439
 - design specifications, coding and, 438–440
 - effective coding, 440
 - evolutionary model issues, 434–435
 - focus, choosing, 434–436
 - initial design decisions, 436–437
 - logic, 440
 - model content issues, 435–436
 - naming conventions, 439–440
 - object plan, 438
 - schedules and assignments, 437–438
 - top-down plan, 437
 - demonstration, SDLC, 18–19
 - evolutionary prototypes, 428, 434–435
 - GB Video example, 440–452
 - prototyping life cycle, 431–432
 - throwaway prototypes, 71–72, 428, 434
 - Prototype-based project planning, 70–72
 - Prototyping, 22–23, 70–72. *see also* Prototype-based POC design
 - life cycle, 431–432
 - prototype-based project planning, 70–72
 - Pseudocode, 392–396
 - flow of control and, 395–396
 - inputting data, 393
 - operating on a data store, 393–394
 - outputting data, 396
 - performing a procedure, 394
 - transforming data, 393
 - Punctuated-equilibrium maturation, of project teams, 38
- ## R
- Ranking methods, outsourced systems, 363–364
 - Rapid application development (RAD), 20. *see also* Rapid development (RD)
 - Rapid development (RD), 20–21
 - project planning model, 68–73
 - SOW approach. *see* Statement of work (SOW)
 - Ratings, for outsourced software, 364–367
 - RCA, 354

- Readability of final documentation, 461
 - Recommendation, 299, 329–332
 - justification of, strategic alignment and, 196
 - slide, in project presentations, 101
 - Recording client meetings, 219
 - Reeves, Laura, 422
 - Referential integrity arrow, 140
 - Refresher training, 467
 - Rehearsals, of project presentations, 101
 - Relational database, 381–382
 - ERD model for, 10
 - Relational model. *see also* Logical data models
 - ER model terms/concepts, compared, 137
 - Relational schema, 140–144
 - box schema, 140
 - column heading schema, 140
 - data structure and system design, 381–382
 - ERDs converted to, 141–142
 - set notation schema, 140
 - for unary relationships, 142–144
 - Relationships
 - entity relationship model, 117, 126–128
 - identification step, simple ERD, 120
 - metadata, 135
 - relationship lines, use case diagrams, 179
 - Removal, of a team member, 54–55
 - Repeating groups, of multivalued attributes, 124
 - Report/documentation tests, at project completion, 457
 - Request for information (RFI), 354
 - Request for proposal (RFP), 18, 72, 341, 354–363
 - bids, 356, 367
 - content of, 355–356
 - defined, 355
 - Request for quote (RFQ), 341, 354
 - Requirements
 - changing, 84, 86–87
 - refining, outsourcing. *see* Outsourcing
 - Resignation, of a team member, 55
 - Resistance to change, 85–86
 - Resource reallocations, 88
 - Respecting clients, 219
 - Response times, 308
 - Retention/change analysis, narrative model, 234–239
 - Retrieving object data (CURD), 177
 - Return on investment (ROI), 14, 322
 - Reusability, object design and, 184
 - Revenue enhancement, 317
 - Review points, 89–90
 - Risk, 313. *see also* Cost/benefit analyses
 - of alternative solutions, 313–315
 - analysis, SDLC, 18
 - reduction, 15, 314–315, 317
 - sponsor risk, 314
 - staff risk, 314
 - Role assignments, 41–42
 - project team code of conduct and, 42
 - for visits to clients, 218
 - workload and, 43
 - Ross, Margy, 422
 - Rumbaugh, James, 156, 190
- S**
- Safe communication, 315
 - Safe design practices, 314
 - Samples, 223
 - SAP, 73, 381, 427
 - Sarson, Trish, 156, 190
 - Satisficing, 69, 391
 - Scenarios, use case diagrams, 180
 - Schedules of completion
 - constructing, 77–80
 - and cost feasibility of alternatives, 312
 - prototype-based POC design, 437–438
 - updating, example, 94
 - Scheduling visits to clients, 217
 - Schneider, G., 10, 31, 190
 - Schwalbe, Kathy, 64, 77, 111
 - Scope of projects
 - in project definition report, 209–210
 - reducing, 88
 - scope creep, 68
 - SDLC. *see* Systems development life cycle (SDLC)
 - Second normal form, logical data models, 144
 - Secretaries, relationships with, 218
 - Secretive genius, 56–57
 - Security
 - feasibility of alternative solutions, 313
 - features in system design, 378
 - of outsourced products, 346
 - Self-management, of project teams, 39
 - Selling, of products and services, 12
 - Sequence constraints, 76–77, 78
 - Sequence diagrams, 26, 183–184
 - Sequential implementation, 464–465
 - Services, selling, 12
 - Set notation schema, 140

- Setup tips, project presentations, 102
- Sharing, of written documents, 51
- Signing and dating
 - communications, 52
 - statements of work (SOW), 81
- Silver, Denise, 70, 111
- Simon, Herbert A., 299, 311, 336
- Simplified, reduced-form ERD (SERD), 130–131
- Simulated POC models, 426
- Sink, 159
- Size of projects, as potential problem, 209
- Skills inventory, 36, 39–41, 46
- Slack, 78
- Slippage, 78
- Slowly changing attributes, 411
- Software licenses, outsourcing and, 347–348
- Software specifications, system design, 378.
 - see also* Features, proposed system
- Solutions. *see* Alternative solutions
- Source, 159
- Sourcing option. *see* Alternative solutions
- SOW. *see* Statement of work (SOW)
- Specialization, of supertypes, 130
- Specifications, 17–18, 19. *see also* Features, proposed system
 - for module design, 391–392
 - slide, project presentations, 100
- Spiral model, 66–67, 429
- Split data flow, in DFDs, 164–165
- Sponsor risk, 314
- Sponsors, 6, 314
- Stability, of vendors, 352
- Staff risk, 314
- Standard query language (SQL), 145–146
- Standards manager, 42
- Start date and end date (due date), 77, 78
 - extensions of due date, 78, 88
- Start up, of project teams, 36–37
- State diagram, 26
- Statement of work (SOW), 70, 80–84
 - client deliverables, 81
 - client resources, 81
 - elements of, 80–81
 - example, 82–83
 - infeasibility of, future, 84
 - project description, 80
 - project schedule, 81
 - project success criteria, 81
 - requirements, changing, 84
 - signatures required, 81
 - work product, 80–81
- Static data, 146
- Static POC model, 424
- Status reports, 87–88
- Storming stage, of team maturation, 38
- Stowe, John D., 336
- Strategic alignment, 8, 195–202
 - for acceptance/credibility, organizational, 196
 - determining alignment, 198–202
 - objectives and goals, identifying, 199–200
 - organizational culture and, 199
 - project contribution, identifying, 200–201
 - project success, 202
 - understanding the organization, 198–199
 - financial sponsorship from, 196
 - ongoing project decisions and, 196
 - for organizational value support, 196
 - the organization case, 196–198
 - the organization, 197
 - the organization's goals and objectives, 197
 - project contribution, 197–198
 - project success criteria, 198
 - project definition report and, 204
 - recommended solution justification and, 196
 - slide, project presentations, 100
- Structural analysis/design, CASE tools for, 67
- Structure, 206
- Structured code, 376
- Structured system acquisition, 14–15
- Stub testing, at project completion, 460
- Subtypes, 129–130
- Successful projects
 - criteria, 202, 205
 - in project definition report, 205
 - statement of work and, 81
 - strategic alignment and, 198, 202
 - critical success measures, 92
 - project teams and, 36–38, 47–48, 92
 - strategic alignment and, 202
 - success index, 210
- Sufficiency rules, DFDs, 164
- Summary slide, project presentations, 101
- Supertypes, 129–130
- Support, from vendors, 352–353
- Supportive behavior, payments for, 50
- Surrogate keys, 409
- Surveys, 198
 - for information collection, 223
- Symbols, 156
 - used in DFDs, 158–159
 - used in ERDs, 117–118

- Synthesis phase, 256. *see also* Features, proposed system
 - System acquisition, 13
 - System boundary, use case diagrams, 179
 - System controls, 471
 - System delivery
 - generation of project plans, 75
 - SDLC, 18–19
 - System design, 6, 373–422. *see also* Systems development life cycle (SDLC)
 - clients, roles in, 379
 - cohesion and, 376, 390
 - data/data design, 375
 - data owners, roles in, 379
 - data structure, specifying, 379–386
 - data item metadata, 383
 - data types, 383
 - metadata, 382–386, 389, 396–399
 - other data schema, 386
 - relational schema, 381–382
 - data warehouse design. *see* Data warehouse
 - dialog-driven. *see* Dialog-driven systems
 - framework for, 375–379
 - hardware specifications, 378
 - inexpensive trial solutions and, 376
 - maintainability and, 376
 - maintainers, roles in, 379
 - module coupling, 376, 390–391
 - module design. *see* Module design
 - network impacts, 378
 - operators, roles in, 379
 - organizational infrastructure and, 375, 378–379
 - parameter confirmation, POC model, 424, 425
 - physical data flow diagrams, 389
 - physical infrastructure and, 375, 377–378
 - process design, 376, 387–389
 - process model metadata, 389
 - program structure charts (PSCs), 387–389
 - security features, 378
 - software specifications, 378
 - structured code, 376
 - users, roles in, 379
 - workload allocation and, 376
 - System operation and maintenance, 19
 - Systems analysis and design. *see* System design
 - Systems development. *see* System design; Systems development life cycle (SDLC)
 - Systems development life cycle (SDLC), 16–19
 - project and team organization, 16–17
 - project definition, 17–18
 - for project planning, 65, 67
 - proposed system, 18
 - prototyping and, 433
 - questions to ask clients, 74–76
 - system delivery, 18–19
 - waterfall concept, 17
 - System solutions. *see* Information system solutions
 - System specification, POC model, 424, 425
- ## T
- Table, 137
 - Table metadata, 383
 - Table of contents, 95
 - Task times/sequence, project plans, 76–77
 - Teams. *see* Project team
 - Team lead, 47
 - Technical complexity, 209
 - Technical feasibility, 312
 - Technology-driven development, 21–22
 - Technology driver, on presentation team, 99
 - Technology risk, 313–315
 - Telephone calls, 51
 - in place of meetings, 218
 - to schedule client visits, 217–218
 - Ternary relationship, 126–128
 - Testing
 - at project completion, 456–461
 - SDLC and, 19
 - Third normal form, logical data models, 144–145
 - Thorntwaite, Warren, 422
 - Throwaway prototypes, 71–72, 428, 434
 - Time-bomb components, 348
 - Time/timing
 - activity/task times and sequence, 76–77
 - arriving at meetings with clients, 219
 - constraints on solutions, 208
 - immediacy, time and, 274
 - MDFDs and, 274
 - of reports, 92
 - respecting, 53
 - response times, 308
 - of tasks, 76–77
 - time-value-of-money methods, 320
 - TIPOT charts, 399, 400–401
 - Title, alternative solutions report, 309
 - Title slide, project presentations, 100
 - Top-down plan, prototype-based POC design, 437
 - Total cost of ownership, 318
 - Total specialization, supertypes, 130
 - Traditional data structure model, 115

Training, 10–11
 education regarding client problems, 217
 follow-on, 466, 467–468
 implementation phase, 465–469
 initial, 466–467
 refresher, 467

Transform (difference reduction) method, 261–262

Transforming data, pseudocode and, 393

Transitive dependency, 144–145

Trial and error, proposed systems and, 259, 260

Triggers, 269–275

Trivial projects, 209

Type attribute, of object classes, 182

U

UML (Uniform Modeling Language), 26

Unary relationship, 126–128
 relational schema for, 142–144

Unified Modeling Language (UML), 26, 156, 178

Unique key, 137

Updated schedules, example, 94

Updating object data (CURD), 177

Usability
 of outsourced products, 345
 POC model, 424, 430
 reusability, object design and, 184

Use case, 26, 179
 diagrams, 178–180, 287–288

Users, 6
 roles of, in system design, 379

V

Valacich, Joseph S., 10, 31, 153, 190, 295

Validity, of POC models, 431

Value Chain Model, 263–264

Vendors, 7
 roles in outsourcing, 351–354
 delivery record, 352, 353
 selecting vendors, 353–354
 vendor features, 352–353
 vendor stability, 352
 vendor support, 352–353
 vendor risk, 313–314

Verbs, active, present-tense, 97

Visibility attribute, of object classes, 182

Vision, of an organization, 198–199

Visiting with clients, 218–220
 preparing for, 216–218

Visual aids, for project presentations, 99–101

W

Walk-through tests, at project completion, 457–459

Waterfall concept, 17

Weak entities, 124–125

Web-based systems, 68

Weber, Ron, 479

Weekly report, 92

Weighted features analysis, 364, 366

Whitten, Jeffrey L., 10, 31, 114, 153, 156, 190, 295

Winters, J. P., 10, 31, 190

Wood, Jane, 70, 111

Working in a team, 51–53
 communication and, 51–52
 manager relations, 52–53

Working with the client. *see* Client(s)

Workloads
 allocation of, 376
 roles and, 42

Work product, statement of work and, 80–81

Wrapper modules, 349

Writer, 42

Written narratives, context-level DFD prepared
 from, 168

Written reports, 92–98
 appearance requirements, 98
 contents of, example, 95
 editing, 97–98
 executive summary, 96
 good writing for, 96–97
 headings and fonts, 95–96
 introduction, 96
 progress report example, 93
 table of contents, 95
 updated schedules, example, 94

Y

Yourdon, Edward, 156, 190, 376–377, 422

Z

Zero-base design option, 301

Zuboff, Shoshana, 295

