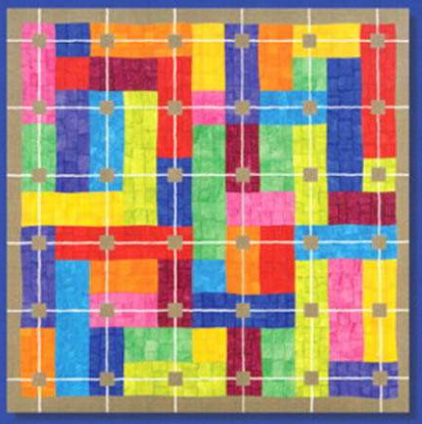
EDITION 8

SYSTEMS ANALYSIS AND DESIGN



KENDALL & KENDALL

SYSTEMS ANALYSIS AND DESIGN

This page intentionally left blank

SYSTEMS ANALYSIS AND DESIGN

EIGHTH EDITION

KENNETH E. KENDALL
Rutgers University
School of Business–Camden
Camden, New Jersey

JULIE E. KENDALL
Rutgers University
School of Business–Camden
Camden, New Jersey

Prentice Hall
Boston Columbus Indianapolis New York San Francisco
Upper Saddle River Amsterdam Cape Town Dubai London Madrid
Milan Munich Paris Montreal Toronto Delhi Mexico City
Sao Paulo Sydney Hong Kong Seoul Singapore Taipei Tokyo

To the memory of Julia A. Kendall and the memory of Edward J. Kendall, whose lifelong example of working together will inspire us forever.

Editorial Director: Sally Yagan
Editor in Chief: Eric Svendsen
Executive Editor: Bob Horan
Editorial Project Manager: Kelly Loftus
Editorial Assistant: Jason Calcano
Director of Marketing: Patrice Lumumba Jones
Senior Marketing Manager: Anne Fahlgren
Senior Managing Editor: Judy Leale
Associate Production Project Manager: Ana Jankowski
Senior Operations Supervisor: Arnold Vila
Senior Art Director: Steve Frim
Text and Cover Designer: Jill Lehan
Cover Art: Richard Kalina, "Carthage" collage, acrylic, flashe on linen: © 2006
Media Project Manager: Lisa Rinaldi
Media Editor: Denise Vaughn
Full-Service Project Management/Composition: S4Carlisle Publishing Services, Inc.
Printer/Binder: Courier/Kendallville
Cover Printer: Lehigh-Phoenix Color/Hagerstown
Text Font: 10/12 Times

"Carthage" © 2006 Richard Kalina, used with permission from the artist.
Credits and acknowledgments borrowed from other sources and reproduced, with permission, in this textbook appear on appropriate page within text.

Microsoft® and Windows® are registered trademarks of the Microsoft Corporation in the U.S.A. and other countries. Screen shots and icons reprinted with permission from the Microsoft Corporation. This book is not sponsored or endorsed by or affiliated with the Microsoft Corporation.

Copyright © 2011, 2008, 2005 by Pearson Education, Inc., publishing as Prentice Hall, One Lake Street, Upper Saddle River, New Jersey 07458. All rights reserved. Manufactured in the United States of America. This publication is protected by Copyright, and permission should be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. To obtain permission(s) to use material from this work, please submit a written request to Pearson Education, Inc., Permissions Department, One Lake Street, Upper Saddle River, New Jersey 07458.

Many of the designations by manufacturers and seller to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed in initial caps or all caps.

Library of Congress Cataloging-in-Publication Data
Kendall, Kenneth E.
Systems analysis and design / Kenneth E. Kendall, Julie E. Kendall. — 8th ed.
p. cm.
Includes bibliographical references and index.
ISBN-13: 978-0-13-608916-2
ISBN-10: 0-13-608916-X
1. System design. 2. System analysis. 3. Systems programming (Computer science)
I. Kendall, Julie E., II. Title.
QA76.9.S88K45 2010
005.4'2—dc22 2009031275

Prentice Hall
is an imprint of
PEARSON

www.pearsonhighered.com

10 9 8 7 6 5 4 3 2 1
ISBN-10: 0-13-608916-X
ISBN-13: 978-0-13-608916-2

COMPANY TRADEMARKS

Apple and Macintosh are registered trademarks of Apple Computer. 1Password is a registered trademark of Agile Web Solutions. Bento is a registered trademark of FileMaker. Dragon NaturallySpeaking is a registered trademark of Nuance. Dreamweaver, Adobe Flash, and FormFlow are registered trademarks of Adobe Systems Incorporated. DEVONagent and DEVONthink Professional Office are registered trademarks of DEVONtechnologies. Firefox is a trademark of the Mozilla Foundation. Freeway Pro is a registered trademark of Softpress Systems. HyperCase is a registered trademark of Raymond J. Barnes, Richard L. Baskerville, Julie E. Kendall, and Kenneth E. Kendall. Microsoft Windows, Microsoft Access, Microsoft Word, Microsoft PowerPoint, Microsoft Project, Microsoft Excel, and Microsoft Visio are registered trademarks of Microsoft Corporation. OmniFocus is a registered trademark of The Omni Group. OmniGraffle and OmniPlan are registered trademarks of The Omni Group. OmniPage is a trademark of Nuance. Palm is a registered trademark of Palm, Inc. ProModel and Service Model are registered trademarks of ProModel Corporation. Things is a registered trademark of Cultured Code. VMware Fusion is a registered trademark of VMware. Visible Analyst is a registered trademark of Visible Systems Corporation. WinFax Pro and Norton Internet Security are registered trademarks of Symantec. Yojimbo is a registered trademark of Bare Bones Software. Other product and company names mentioned herein may be the trademarks of their respective owners. Companies, names, and/or data used in screens and sample output are fictitious unless otherwise noted.

BRIEF CONTENTS

PART I SYSTEMS ANALYSIS FUNDAMENTALS

1 SYSTEMS, ROLES, AND DEVELOPMENT METHODOLOGIES 1
2 UNDERSTANDING AND MODELING ORGANIZATIONAL SYSTEMS 24
3 PROJECT MANAGEMENT 56

PART II INFORMATION REQUIREMENTS ANALYSIS

4 INFORMATION GATHERING: INTERACTIVE METHODS 103
5 INFORMATION GATHERING: UNOBTRUSIVE METHODS 131
6 AGILE MODELING AND PROTOTYPING 155

PART III THE ANALYSIS PROCESS

7 USING DATA FLOW DIAGRAMS 193
8 ANALYZING SYSTEMS USING DATA DICTIONARIES 228
9 PROCESS SPECIFICATIONS AND STRUCTURED DECISIONS 259
10 OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN USING UML 281

PART IV THE ESSENTIALS OF DESIGN

11 DESIGNING EFFECTIVE OUTPUT 329
12 DESIGNING EFFECTIVE INPUT 371
13 DESIGNING DATABASES 403
14 HUMAN-COMPUTER INTERACTION 441

PART V QUALITY ASSURANCE AND IMPLEMENTATION

15 DESIGNING ACCURATE DATA ENTRY PROCEDURES 485
16 QUALITY ASSURANCE AND IMPLEMENTATION 515

GLOSSARY 557
ACRONYMS 565
INDEX 566

CONTENTS

PART I SYSTEMS ANALYSIS FUNDAMENTALS

1 SYSTEMS, ROLES, AND DEVELOPMENT METHODOLOGIES 1

Types of Systems 2
Transaction Processing Systems 2 / Office Automation Systems and Knowledge Work Systems 2 / Management Information Systems 3 / Decision Support Systems 3 / Artificial Intelligence and Expert Systems 3 / Group Decision Support Systems and Computer-Supported Collaborative Work Systems 3 / Executive Support Systems 4

Integrating Technologies for Systems 4
Ecommerce Applications and Web Systems 4 / Enterprise Systems 5 / Systems for Wireless and Mobile Devices 5 / Open Source Software 5

Need for Systems Analysis and Design 6

Roles of the Systems Analyst 6
Systems Analyst as Consultant 6 / Systems Analyst as Supporting Expert 6

Consulting Opportunity 1.1 *Healthy Hiring: Ecommerce Help Wanted* 7
Systems Analyst as Agent of Change 7 / Qualities of the Systems Analyst 8

The Systems Development Life Cycle 8
Incorporating Human-Computer Interaction Considerations 9 / Identifying Problems, Opportunities, and Objectives 9 / Determining Human Information Requirements 10 / Analyzing System Needs 10 / Designing the Recommended System 11 / Developing and Documenting Software 11 / Testing and Maintaining the System 11 / Implementing and Evaluating the System 11

MAC APPEAL 12
The Impact of Maintenance 12

Using Case Tools 14

The Agile Approach 14
Developmental Process for an Agile Project 15

Object-Oriented Systems Analysis and Design 17

Choosing Which Systems Development Method to Use 19
SUMMARY 19
HYPERCASE® EXPERIENCE 1 20
KEYWORDS AND PHRASES 21
REVIEW QUESTIONS 21
SELECTED BIBLIOGRAPHY 21
CPU CASE EPISODE 1: *The Case Opens* 23

2 UNDERSTANDING AND MODELING ORGANIZATIONAL SYSTEMS 24

Organizations as Systems 24
Interrelatedness and Interdependence of Systems 25

Consulting Opportunity 2.1 ***The E in Vitamin E Stands for Ecommerce*** **26**
Virtual Organizations and Virtual Teams 26 / Taking a Systems Perspective 27 / Enterprise Systems: Viewing the Organization as a System 28

Depicting Systems Graphically 29
Systems and the Context-Level Data Flow Diagram 29 / Systems and the Entity-Relationship Model 30

MAC APPEAL 35

Use Case Modeling 35
Use Case Symbols 36 / Use Case Relationships 36 / Developing System Scope 38 / Developing Use Case Diagrams 38 / Developing Use Case Scenarios 38 / Use Case Levels 39 / Creating Use Case Descriptions 43 / Why Use Case Diagrams Are Helpful 43

Levels of Management 43

Consulting Opportunity 2.2 ***Where There's Carbon, There's a Copy*** **44**
Implications for Information Systems Development 45

Organizational Culture 45

Consulting Opportunity 2.3 ***Pyramid Power*** **46**
SUMMARY 46
HYPERCASE® EXPERIENCE 2 47
KEYWORDS AND PHRASES 48
REVIEW QUESTIONS 48
PROBLEMS 48
GROUP PROJECTS 49
SELECTED BIBLIOGRAPHY 50

CPU CASE EPISODE 2: ***Picturing the Relationships*** **51**

3 PROJECT MANAGEMENT 56

Project Initiation 56
Problems in the Organization 57 / Defining the Problem 57

Consulting Opportunity 3.1 ***The Sweetest Sound I've Ever Sipped*** **58**
Selection of Projects 61

Determining Feasibility 62
Determining Whether It Is Possible 62

Ascertaining Hardware and Software Needs 63
Inventorying Computer Hardware 64 / Estimating Workloads 64 / Evaluating Computer Hardware 65 / Acquisition of Computer Equipment 66 / Software Evaluation 68

Consulting Opportunity 3.2 ***Veni, Vidi, Vendi, or, I Came, I Saw, I Sold*** **70**

Identifying, Forecasting, and Comparing Costs and Benefits 72
Forecasting 72 / Identifying Benefits and Costs 72

Consulting Opportunity 3.3 ***We're Off to See the Wizards*** **73**
Comparing Costs and Benefits 74

Activity Planning and Control 77
Estimating Time Required 77

Consulting Opportunity 3.4 *Food for Thought* 78
Using Gantt Charts for Project Scheduling 79 / Using PERT Diagrams 80

Managing the Project 82
Addressing System Complexity 82

MAC APPEAL 83

Managing Analysis and Design Activities 83
Assembling a Team 83
HYPERCASE® EXPERIENCE 3.1 84
Communication Strategies for Managing Teams 84

Consulting Opportunity 3.5 *Goal Tending* 85
Setting Project Productivity Goals 85 / Motivating Project Team Members 86 / Managing Ecommerce Projects 86 / Creating the Project Charter 87 / Avoiding Project Failures 87

The Systems Proposal 88
Organizing the Systems Proposal 88 / Using Figures for Effective Communication 89
SUMMARY 91
HYPERCASE® EXPERIENCE 3.2 92
KEYWORDS AND PHRASES 93
REVIEW QUESTIONS 93
PROBLEMS 94
GROUP PROJECTS 98
SELECTED BIBLIOGRAPHY 98

CPU CASE EPISODE 3: *Getting to Know U* 100

PART II INFORMATION REQUIREMENTS ANALYSIS

4 INFORMATION GATHERING: INTERACTIVE METHODS 103

Interviewing 103
Five Steps in Interview Preparation 104 / Question Types 105 / Arranging Questions in a Logical Sequence 107

Consulting Opportunity 4.1 *Strengthening Your Question Types* 108
Writing the Interview Report 109

Consulting Opportunity 4.2 *Skimming the Surface* 110

Joint Application Design 111
Conditions That Support the Use of JAD 111 / Who Is Involved? 111
HYPERCASE® EXPERIENCE 4.1 112
Where to Hold JAD Meetings 112

Consulting Opportunity 4.3 *A Systems Analyst, I Presume?* 113
Accomplishing a Structured Analysis of Project Activities 113 / Potential Benefits of Using JAD in Place of Traditional Interviewing 113 / Potential Drawbacks of Using JAD 113

Using Questionnaires 114
Planning for the Use of Questionnaires 114 / Writing Questions 115 / Using Scales in Questionnaires 118 / Designing the Questionnaires 119

Consulting Opportunity 4.4 *The Unbearable Questionnaire* 120

Consulting Opportunity 4.5 *Order in the Courts* 121
Administering Questionnaires 122
SUMMARY 122

HYPERCASE® EXPERIENCE 4.2 123
KEYWORDS AND PHRASES 123
REVIEW QUESTIONS 124
PROBLEMS 124
GROUP PROJECTS 127
SELECTED BIBLIOGRAPHY 127

CPU CASE EPISODE 4: *I'll Listen Now, Ask Questions Later* 128

5 INFORMATION GATHERING: UNOBTRUSIVE METHODS 131

Sampling 131
The Need for Sampling 132 / Sampling Design 132 / The Sample Size Decision 134

Consulting Opportunity 5.1 *Trapping a Sample* 135

Investigation 136
Analyzing Quantitative Documents 136

Consulting Opportunity 5.2 *A Rose by Any Other Name . . . Or Quality, Not Quantities* 137
Analyzing Qualitative Documents 140
HYPERCASE® EXPERIENCE 5.1 141

Observing a Decision Maker's Behavior 142
Observing a Typical Manager's Decision-Making Activities 142

Observing the Physical Environment 142
Structured Observation of the Environment (STROBE) 142

Consulting Opportunity 5.3 *Don't Bank on Their Self-Image or Not Everything Is Reflected in a Mirror* 145
Applying Strobe 146

MAC APPEAL 147
SUMMARY 148
HYPERCASE® EXPERIENCE 5.2 149
KEYWORDS AND PHRASES 150
REVIEW QUESTIONS 150
PROBLEMS 150
GROUP PROJECTS 152
SELECTED BIBLIOGRAPHY 153

CPU CASE EPISODE 5: *Seeing Is Believing* 154

6 AGILE MODELING AND PROTOTYPING 155

Prototyping 156
Kinds of Prototypes 156 / Prototyping as an Alternative to the SDLC 157

Developing a Prototype 158

Consulting Opportunity 6.1 *Is Prototyping King?* 159
Guidelines for Developing a Prototype 159

Consulting Opportunity 6.2 *Clearing the Way for Customer Links* 160
Disadvantages of Prototyping 160

Consulting Opportunity 6.3 *To Hatch a Fish* 161
Advantages of Prototyping 161 / Prototyping Using COTS Software 161

Consulting Opportunity 6.4 *This Prototype Is All Wet* 162
Users' Role in Prototyping 162

Rapid Application Development 163
Phases of RAD 164 / Comparing RAD to the SDLC 165

Agile Modeling 166
Values and Principles of Agile Modeling 166 / Activities, Resources, and Practices of Agile Modeling 168 / The Agile Development Process 171

MAC APPEAL 173
Lessons Learned from Agile Modeling 175

Comparing Agile Modeling and Structured Methods 176
Improving Efficiency in Knowledge Work: SDLC Versus Agile 177 / Risks Inherent in Organizational Innovation 179
SUMMARY 181
HYPERCASE® EXPERIENCE 6 182
KEYWORDS AND PHRASES 183
REVIEW QUESTIONS 183
PROBLEMS 183
GROUP PROJECTS 185
SELECTED BIBLIOGRAPHY 185

CPU CASE EPISODE 6: *Reaction Time* 186

PART III THE ANALYSIS PROCESS

7 USING DATA FLOW DIAGRAMS 193

The Data Flow Approach to Human Requirements Determination 193
Advantages of the Data Flow Approach 193 / Conventions Used in Data Flow Diagrams 194

Developing Data Flow Diagrams 195
Creating the Context Diagram 195 / Drawing Diagram 0 (The Next Level) 196 / Creating Child Diagrams (More Detailed Levels) 198 / Checking the Diagrams for Errors 198

Logical and Physical Data Flow Diagrams 200
Developing Logical Data Flow Diagrams 202 / Developing Physical Data Flow Diagrams 203 / Partitioning Data Flow Diagrams 206

A Data Flow Diagram Example 207
Developing the List of Business Activities 207 / Creating a Context-Level Data Flow Diagram 208 / Drawing Diagram 0 210 / Creating a Child Diagram 211 / Creating a Physical Data Flow Diagram from the Logical DFD 212 / Partitioning the Physical DFD 213

Partitioning Web Sites 213

Consulting Opportunity 7.1 *There's No Business Like Flow Business* 216

Communicating Using Data Flow Diagrams 217
SUMMARY 217
HYPERCASE® EXPERIENCE 7 218
KEYWORDS AND PHRASES 218
REVIEW QUESTIONS 219
PROBLEMS 219
GROUP PROJECTS 221
SELECTED BIBLIOGRAPHY 221

CPU CASE EPISODE 7: *Just Flowing Along* 222

8 ANALYZING SYSTEMS USING DATA DICTIONARIES 228

The Data Dictionary 228
Need for Understanding the Data Dictionary 229

The Data Repository 229
Defining the Data Flows 230 / Describing Data Structures 231 / Logical and Physical Data Structures 233 / Data Elements 234 / Data Stores 236

Creating the Data Dictionary 238
Analyzing Input and Output 239

Consulting Opportunity 8.1 ***Want to Make It Big in the Theatre? Improve Your Diction(ary)!*** **240**
Developing Data Stores 241

Using the Data Dictionary 242
Using Data Dictionaries to Create XML 243 / XML Document Type Definitions 244 / XML Schemas 246

HYPERCASE® EXPERIENCE 8 247

SUMMARY 248
KEYWORDS AND PHRASES 248
REVIEW QUESTIONS 248
PROBLEMS 249
GROUP PROJECTS 251
SELECTED BIBLIOGRAPHY 251

CPU CASE EPISODE 8: ***Defining What You Mean*** **252**

9 PROCESS SPECIFICATIONS AND STRUCTURED DECISIONS 259

Overview of Process Specifications 259
Process Specification Format 260

Structured English 261
Writing Structured English 261

Consulting Opportunity 9.1 ***Kit Chen Kaboodle, Inc.*** **263**

Consulting Opportunity 9.2 ***Kneading Structure*** **264**
Data Dictionary and Process Specifications 265

Decision Tables 266
Developing Decision Tables 267

Consulting Opportunity 9.3 ***Saving a Cent on Citron Car Rental*** **269**
Checking for Completeness and Accuracy 270

Decision Trees 271

Consulting Opportunity 9.4 ***A Tree for Free*** **272**
Drawing Decision Trees 272

Choosing a Structured Decision Analysis Technique 273

SUMMARY 273
HYPERCASE® EXPERIENCE 9 274
KEYWORDS AND PHRASES 274
REVIEW QUESTIONS 274
PROBLEMS 274
GROUP PROJECTS 276
SELECTED BIBLIOGRAPHY 276

CPU CASE EPISODE 9: ***Tabling a Decision*** **277**

10 OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN USING UML 281

Object-Oriented Concepts 282
Objects 282 / Classes 282 / Inheritance 283

Consulting Opportunity 10.1 *Around the World in 80 Objects* 284

CRC Cards and Object Think 284
Interacting During a CRC Session 284

The Unified Modeling Language (UML) Concepts and Diagrams 286

Use Case Modeling 287

Activity Diagrams 290
Creating Activity Diagrams 292

Consulting Opportunity 10.2 *Recycling the Programming Environment* 293
Repository Entries for an Activity Diagram 294

Sequence and Communication Diagrams 294
Sequence Diagrams 294 / Communication Diagrams 296

Class Diagrams 297
Method Overloading 298 / Types of Classes 299 / Defining Messages and Methods 300

Enhancing Sequence Diagrams 300
A Class Example for the Web 300 / Presentation, Business, and Persistence Layers in Sequence Diagrams 302

Enhancing Class Diagrams 303
Relationships 304 / Generalization/Specialization (Gen/Spec) Diagrams 306

Statechart Diagrams 309
A State Transition Example 310

Packages and Other UML Artifacts 311

Consulting Opportunity 10.3 *Developing a Fine System That Was Long Overdue: Using Object-Oriented Analysis for the Ruminski Public Library System* 313

Putting UML to Work 313

Consulting Opportunity 10.4 *C-Shore++* 315

The Importance of Using UML for Modeling 315

SUMMARY 316
HYPERCASE® EXPERIENCE 10 317
KEYWORDS AND PHRASES 317
REVIEW QUESTIONS 318
PROBLEMS 318
SELECTED BIBLIOGRAPHY 319

CPU CASE EPISODE 10: *Classy Objects* 320

PART IV THE ESSENTIALS OF DESIGN

11 DESIGNING EFFECTIVE OUTPUT 329

Output Design Objectives 329
Designing Output to Serve the Intended Purpose 329 / Designing Output to Fit the User 330 / Delivering the Appropriate Quantity of Output 330 / Making Sure the Output Is Where It Is Needed 330 / Providing the Output on Time 330 / Choosing the Right Output Method 330

Relating Output Content to Output Method 330
Output Technologies 331

Consulting Opportunity 11.1 *Your Cage or Mine?* 335
Factors to Consider When Choosing Output Technology 336

Consulting Opportunity 11.2 *A Right Way, a Wrong Way, and a Subway* 339

Realizing How Output Bias Affects Users 340
Recognizing Bias in the Way Output Is Used 340 / Avoiding Bias in the Design of Output 341 / Designing Printed Output 341 / Guidelines for Printed Report Design 341

Consulting Opportunity 11.3 *Should This Chart Be Barred?* 342

Designing Output for Displays 344
Guidelines for Display Design 344 / Using Graphical Output in Screen Design 345 / Dashboards 346 / Widgets and Gadgets—Changing the Desktop Metaphor 347

Consulting Opportunity 11.4 *Is Your Work a Grind?* 348

Designing a Web Site 348
General Guidelines for Designing Web Sites 348

MAC APPEAL 351

Consulting Opportunity 11.5 *A Field Day* 356
Maintaining Web Sites 356 / Creating Blogs (Web Logs) 357

Output Production and XML 357
Ajax 358

HYPERCASE® EXPERIENCE 11 360

SUMMARY 360
KEYWORDS AND PHRASES 361
REVIEW QUESTIONS 361
PROBLEMS 362
GROUP PROJECTS 365
SELECTED BIBLIOGRAPHY 365

CPU CASE EPISODE 11: *Reporting on Outputs* 366

12 DESIGNING EFFECTIVE INPUT 371

Good Form Design 371
Making Forms Easy to Fill In 372 / Meeting the Intended Purpose 374 / Ensuring Accurate Completion 375 / Keeping Forms Attractive 375 / Controlling Business Forms 376

Good Display and Web Forms Design 376
Keeping the Display Simple 376

Consulting Opportunity 12.1 *This Form May Be Hazardous to Your Health* 377
Keeping the Display Consistent 378 / Facilitating Movement 378 / Designing an Attractive and Pleasing Display 378 / Using Icons in Display Design 378

Consulting Opportunity 12.2 *Squeezin' Isn't Pleasin'* 379
Graphical User Interface Design 379 / Form Controls and Values 381 / Hidden Fields 382 / Event-Response Charts 382 / Dynamic Web Pages 383 / Three-Dimensional Web Pages 385 / Ajax (Asynchronous JavaScript and XML) 387 / Using Color in Display Design 389

Intranet and Internet Page Design 389

Consulting Opportunity 12.3 *It's Only Skin Deep* 390

MAC APPEAL 392

SUMMARY 392
HYPERCASE® EXPERIENCE 12 393
KEYWORDS AND PHRASES 394
REVIEW QUESTIONS 394
PROBLEMS 395
GROUP PROJECTS 397
SELECTED BIBLIOGRAPHY 398

CPU CASE EPISODE 12: *Forming Screens and Screening Forms* 399

13 DESIGNING DATABASES 403

Databases 403

Consulting Opportunity 13.1 *Hitch Your Cleaning Cart to a Star* 404

Data Concepts 404
Reality, Data, and Metadata 404 / Files 410 / Relational Databases 411

Normalization 413
The Three Steps of Normalization 413 / A Normalization Example 414 / Using the Entity-Relationship Diagram to Determine Record Keys 421 / One-to-Many Relationship 422 / Many-to-Many Relationship 423

Guidelines for Master File/Database Relation Design 424
Integrity Constraints 424

MAC APPEAL 425
Anomalies 425

Making Use of the Database 426
Steps in Retrieving and Presenting Data 426

Denormalization 426

Data Warehouses 429
Online Analytic Processing 429 / Data Mining 429

Consulting Opportunity 13.2 *Storing Minerals for Health, Data for Mining* 430

Consulting Opportunity 13.3 *Losing Prospects* 431
SUMMARY 432
HYPERCASE® EXPERIENCE 13 433
KEYWORDS AND PHRASES 433
REVIEW QUESTIONS 434
PROBLEMS 434
GROUP PROJECT 436
SELECTED BIBLIOGRAPHY 436

CPU CASE EPISODE 13: *Back to Data Basics* 437

14 HUMAN-COMPUTER INTERACTION 441

Understanding Human-Computer Interaction 441
How Fit Affects Performance and Well-Being 442 / The Technology Acceptance Model and Attitude 443

Usability 444
Designing for the Cognitive Styles of Individual Users 444 / Physical Considerations in HCI Design 448 / Considering Human Limitations, Disabilities, and Design 449

Consulting Opportunity 14.1 *School Spirit Comes in Many Sizes* 450
Implementing Good HCI Practices 450

Types of User Interface 451
Natural-Language Interfaces 451 / Question-and-Answer Interfaces 452 / Menus 453

Consulting Opportunity 14.2 *I'd Rather Do It Myself* 454
Form-Fill Interfaces (Input/Output Forms) 454 / Command-Language Interfaces 455

Consulting Opportunity 14.3 *Don't Slow Me Down* 456
Graphical User Interfaces 456

Consulting Opportunity 14.4 *That's Not a Lightbulb* 457
Other User Interfaces 457

Guidelines for Dialog Design 458
Meaningful Communication 459 / Minimal User Action 459

Consulting Opportunity 14.5 *Waiting to Be Fed* 461
Standard Operation and Consistency 461

Feedback for Users 461
Types of Feedback 462 / Including Feedback in Design 464

Special Design Considerations for Ecommerce 465
Soliciting Feedback from Ecommerce Web Site Customers 465 / Easy Navigation for Ecommerce Web Sites 465

Consulting Opportunity 14.6 *When You Run a Marathon,It Helps to Know Where You're Going* 466

MAC APPEAL 467

Mashups 468

Designing Queries 468
Query Types 469 / Query Methods 471

Consulting Opportunity 14.7 *Hey, Look Me Over (Reprise)* 472
SUMMARY 474
KEYWORDS AND PHRASES 474
HYPERCASE® EXPERIENCE 14 475
REVIEW QUESTIONS 476
PROBLEMS 476
GROUP PROJECTS 477
SELECTED BIBLIOGRAPHY 478

CPU CASE EPISODE 14: *Up to the Users* 479

PART V QUALITY ASSURANCE AND IMPLEMENTATION

15 DESIGNING ACCURATE DATA ENTRY PROCEDURES 485

Effective Coding 485
Keeping Track of Something 486 / Classifying Information 487 / Concealing Information 488 / Revealing Information 488 / Unicode 490 / Requesting Appropriate Action 491 / General Guidelines for Coding 491

Consulting Opportunity 15.1 *It's a Wilderness in Here* 492

Consulting Opportunity 15.2 *Catching a Summer Code* 494

Effective and Efficient Data Capture 494
Deciding What to Capture 494 / Letting the Computer Do the Rest 495 / Avoiding Bottlenecks and Extra Steps 496 / Starting with a Good Form 496 / Choosing a Data Entry Method 496

Consulting Opportunity 15.3 *To Enter or Not to Enter: That Is the Question* 499

Ensuring Data Quality Through Input Validation 499
Validating Input Transactions 500 / Validating Input Data 500

Consulting Opportunity 15.4 *Do You Validate Parking?* 504
The Process of Validation 505

Accuracy Advantages in Ecommerce Environments 506
Customers Keying Their Own Data 506 / Storing Data for Later Use 506 / Using Data Through the Order Fulfillment Process 506 / Providing Feedback to Customers 506
HYPERCASE® EXPERIENCE 15 507
SUMMARY 507
KEYWORDS AND PHRASES 508
REVIEW QUESTIONS 508
PROBLEMS 509
GROUP PROJECTS 511
SELECTED BIBLIOGRAPHY 511

CPU CASE EPISODE 15: *Entering Naturally* 512

16 QUALITY ASSURANCE AND IMPLEMENTATION 515

The Total Quality Management Approach 516
Six Sigma 516 / Responsibility for Total Quality Management 516 / Structured Walkthrough 517

Consulting Opportunity 16.1 *The Quality of MIS Is Not Strained* 518
Top-Down Systems Design and Development 518

MAC APPEAL 520
Using Structure Charts to Design Modular Systems 520 / Service-Oriented Architecture (SOA) 522

Documentation Approaches 523
Procedure Manuals 523 / The FOLKLORE Method 523

Consulting Opportunity 16.2 *Write Is Right* 524
HYPERCASE® EXPERIENCE 16.1 525
Choosing a Design and Documentation Technique 526

Testing, Maintenance, and Auditing 526
The Testing Process 526

Consulting Opportunity 16.3 *Cramming for Your Systems Test* 528
Maintenance Practices 528 / Auditing 529

Implementing Distributed Systems 529
Client-Server Technology 529 / Cloud Computing 531 / Network Modeling 533

Training Users 536
Training Strategies 537 / Guidelines for Training 537

Consulting Opportunity 16.4 *You Can Lead a Fish to Water . . . But You Can't Make It Drink* 538

Conversion to a New System 539
Conversion Strategies 539 / Other Conversion Considerations 540 / Organizational Metaphors and Their Relationship to Successful Systems 541

Security Concerns for Traditional and Web-Based Systems 542
Physical Security 542 / Logical Security 542 / Behavioral Security 543 / Special Security Considerations for Ecommerce 543 / Privacy Considerations for Ecommerce 544 / Disaster Recovery Planning 544

Consulting Opportunity 16.5 *The Sweet Smell of Success* 546

Evaluation 546
Evaluation Techniques 546 / The Information System Utility Approach 546

Consulting Opportunity 16.6 *Mopping Up with the New System* 548

Evaluating Corporate Web Sites 548
SUMMARY 550
HYPERCASE® EXPERIENCE 16.2 551
KEYWORDS AND PHRASES 551
REVIEW QUESTIONS 551
PROBLEMS 552
GROUP PROJECTS 554
SELECTED BIBLIOGRAPHY 554

CPU CASE EPISODE 16: *Semper Redundate* 555

GLOSSARY 557

ACRONYMS 565

INDEX 566

PREFACE

NEW TO THIS EDITION

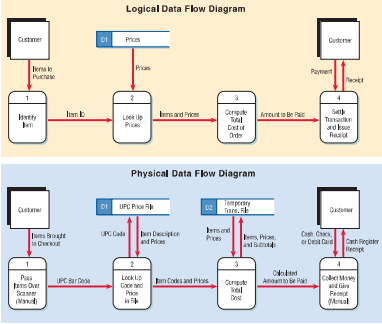
The eighth edition of Kendall & Kendall: *Systems Analysis and Design* includes substantial changes mandated by the rapid changes in the IS field in the last three years, and they are included as a response to the input of our adopters, students, and reviewers. Many new and updated features are integrated throughout. In particular:

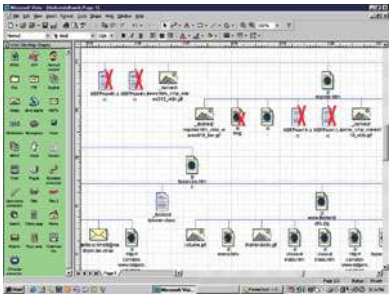
- Early introduction of three methodologies: SDLC, agile approaches, and object-oriented systems analysis and design with UML, along with reasons and situations for when to use them (Chapter 1)
- Expanded coverage of COTS due to their increasing use in organizations (Chapter 3)
- Object-oriented analysis and design expanded and featured in the center of the text so it can be more easily incorporated into term projects (Chapter 10)
- New object-oriented analysis and design problems included in the O-O CPU Episode (Chapter 10)
- New feature throughout the text called *Mac Appeal,* which discusses innovative software available on Macs
- New coverage of cloud computing (Chapter 16)
- New section on SOA (service-oriented architecture) (Chapter 16)
- New end-of-chapter Review Questions, Problems, and Group Exercises
- Reorganized, updated, and strengthened chapter on agile methods (Chapter 6)
- Strengthened project management by introducing the project charter early in the process (Chapter 3)
- Early introduction of how to prepare the systems proposal (Chapter 3)
- Updated and streamlined *CPU Case Episodes*, the case that runs throughout the text. Students can use Microsoft Access, Microsoft Visio, or Visible Analyst to complete new assignments throughout the text
- Updated scenarios, problems, and graphics throughout the text to accompany HyperCase 2.8, a graphical simulation for the Web that allows students to apply their new skills

DESIGN FEATURES

Figures take on a stylized look in order to help students more easily grasp the subject matter.

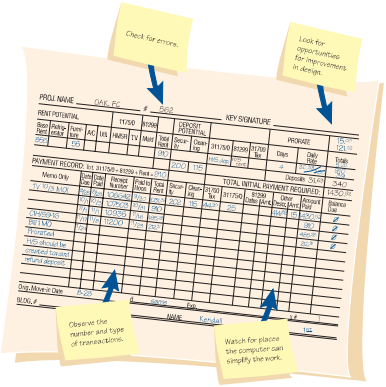
Conceptual diagrams are used to introduce the many tools that systems analysts have at their disposal. This example shows the differences between logical data flow diagrams and physical data flow diagrams. Conceptual diagrams are color coded so that students can distinguish easily among them, and their functions are clearly indicated. Many other important tools are illustrated, including use case diagrams, sequence diagrams, and class diagrams.





Computer displays demonstrate important software features that are useful to the analyst. This example shows how a Web site can be evaluated for broken links by using a package such as Microsoft Visio. Actual screen shots show important aspects of design. Analysts are continuously seeking to improve the appearance of the screens and Web pages they design. Colorful examples help to illustrate why some screen designs are particularly effective.

Paper forms are used throughout to show input and output design as well as the design of questionnaires. Blue ink is always used to show writing or data input, thereby making it easier to identify what was filled in by users. Although most organizations have computerization of manual processes as their eventual goal, much data capture is still done using paper forms. Improved form design enables analysts to ensure accurate and complete input and output. Better forms can also help streamline new internal workflows that result from newly automated business-to-consumer (B2C) applications for ecommerce on the Web.



Tables are used when an important list needs special attention, or when information needs to be organized or classified. In addition, tables are used to supplement the understanding of the reader in a way that departs from how material is organized in the narrative portion of the text. Most analysts find tables a useful way to organize numbers and text into a meaningful "snapshot."

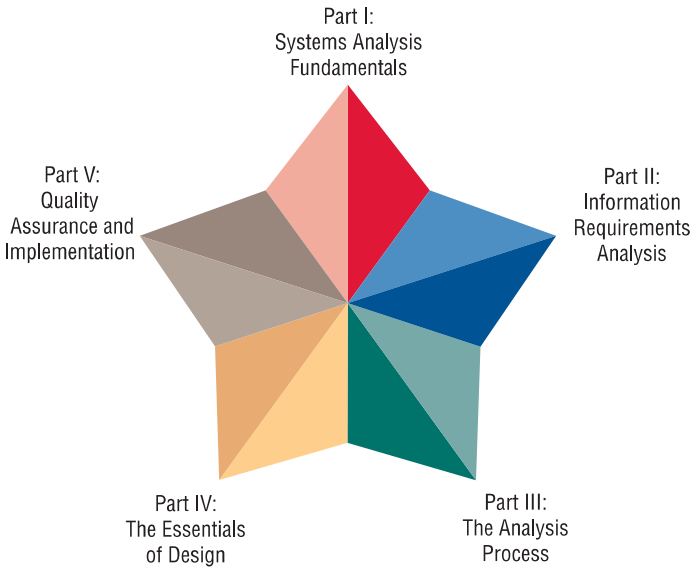
This example of a table from Chapter 3 shows how analysts can refine their activity plans for analysis by breaking them down into smaller tasks and then estimating how much time it will take to complete them. The underlying philosophy of our book is that systems analysis and design is a process that integrates the use of many tools with the unique talents of the systems analyst to systematically improve business through the implementation or modification of computerized information systems. Systems analysts can grow in their work by taking on new IT challenges and keeping current in their profession through the application of new techniques and tools.

Activity	Detailed Activity	Weeks Required
Data gathering	Conduct interviews	3
	Administer questionnaires	4
	Read company reports	4
	Introduce prototype	5
	Observe reactions to prototype	3
Data flow and decision analysis	Analyze data flow	8
Proposal preparation	Perform cost-benefit analysis	3
	Prepare proposal	2
	Present proposal	2

Break these down further.

then estimate time required.

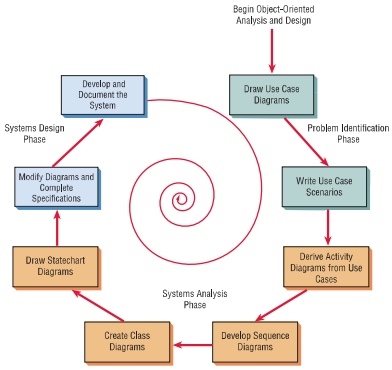
A BRIEF TOUR OF THE EIGHTH EDITION



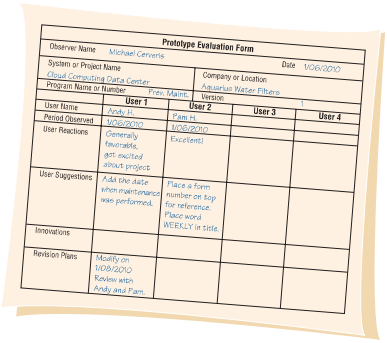
Systems analysis and design is typically taught in one or two semesters. Our book may be used in either situation. The text is appropriate for undergraduate (junior or senior) curricula at a four-year university, graduate school, or community college. The level and length of the course can be varied and supplemented by using real-world projects, HyperCase, or other materials available on the Instructor Resource Center.

The text is divided into five major parts: Systems Analysis Fundamentals (Part I), Information Requirements Analysis (Part II), The Analysis Process (Part III), The Essentials of Design (Part IV), and Quality Assurance and Implementation (Part V).

Part I (Chapters 1–3) stresses the basics that students need to know about what an analyst does and introduces the three main methodologies of the systems development life cycle (SDLC), agile approaches, and object-oriented analysis with UML, along with reasons and situations for when to use them. Part I shows how a variety of emerging information systems, including mobile and wireless technologies, and enterprise systems integrate IT and fit into organizations; how to determine whether a systems project is worthy of commitment; ecommerce project management; and how to manage a systems project using special software tools. The three roles of the systems analyst as consultant, supporting expert, and agent of change are introduced, and ethical issues and professional guidelines for serving as a systems consultant are covered. There is also material on virtual teams and virtual organizations, and the concept of HCI is introduced. The use of open source software (OSS) is also introduced. Chapter 2 includes how to initially approach an organization by drawing context-level data flow diagrams, using entity-relationship models, and developing use cases and use case scenarios. Chapter 3 introduces expanded material on creating the project charter and introduces writing the systems proposal early in the process, no matter what method of analysis and design has been chosen. Expanded coverage of evaluating software and hardware, and when to use COTS (commercial off-the-shelf software), is included. This chapter teaches several methods for forecasting costs and benefits, which are necessary to the discussion of acquiring software and hardware. Chapter 3 helps students evaluate software by assessing trade-offs among creating custom software, purchasing commercial-off-the-shelf (COTS) software, or outsourcing to an application service provider (ASP). Creating a problem definition and determining feasibility are also covered. Chapter 3 guides students in professionally writing and presenting an effective systems proposal, one that incorporates figures and graphs to communicate with users.

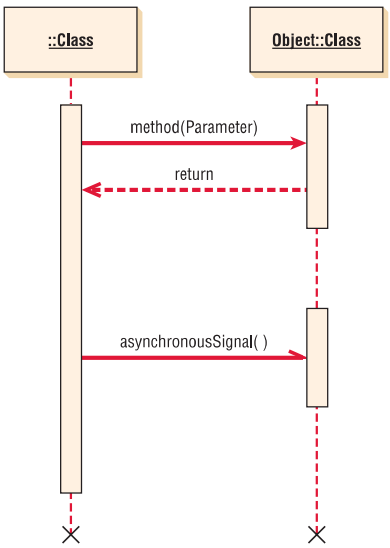


Part II (Chapters 4–6) emphasizes the use of systematic and structured methodologies for performing information requirements analysis. Attention to analysis helps analysts ensure that they are addressing the correct problem before designing the system. Chapter 4 introduces a group of interactive methods, including interviewing, joint application design (JAD), and constructing questionnaires. Chapter 5 introduces a group of unobtrusive methods for ascertaining information requirements of users. These methods include sampling, investigating hard and archival data, and observation of decision makers' behavior and their physical environment. Chapter 6 on agile modeling and prototyping is innovative in its treatment of prototyping as another data-gathering technique that enables the analyst to solve the right problem by getting users involved from the start. Agile approaches have their roots in prototyping, so this chapter begins with prototyping to provide a proper context for understanding, and then takes up the agile approach. The values and principles, activities, resources, practices, processes, and tools associated with agile methodologies are presented. This chapter also includes material on rapid application development (RAD) for human information-requirements gathering and interface design.



Prototype Evaluation Form

Observer Name: Michael Cerveris			Date: 10/6/2010	
System or Project Name: Cloud Computing Data Center		Company or Location: Aquarius Water Filters		
Program Name or Number: Prev. Maint.		Version:		
User Name	User 1: Andy P.	User 2: Pam R.	User 3	User 4
Period Observed	10/6/2010	10/6/2010		
User Reactions	Generally favorable, get excited about project.	Excellent!		
User Suggestions	Add the date when maintenance was performed.	Place a form number on top for reference. Place word WEEKLY in title.		
Innovations				
Revision Plans	Modify on 10/8/2010. Review with Andy and Pam.			



Part III (Chapters 7–10) details the analysis process. It builds on the previous two parts to move students into analysis of data flows as well as structured and semistructured decisions. It provides step-by-step details on how to use structured techniques to draw data flow diagrams (DFDs). Chapter 7 provides coverage of how to create child diagrams; how to develop both logical and physical data flow diagrams; and how to partition data flow diagrams. Chapter 8 features material on the data repository and vertical balancing of data flow diagrams. Chapter 8 also includes extensive coverage of extensible markup language (XML) and demonstrates how to use data dictionaries to create XML. Chapter 9 includes material on developing process specifications. A discussion of both logical and physical process specifications shows how to use process specifications for horizontal balancing. Chapter 9 also covers how to diagram structured decisions with the use of structured English, decision tables, and decision trees. In addition, the chapter covers how to choose an appropriate decision analysis method for analyzing structured decisions and creating process specifications. Push technologies are also introduced.

Part III concludes with Chapter 10 on object-oriented systems analysis and design. This chapter includes an in-depth section on using unified modeling language (UML). There is detailed coverage of the use case model, creating the class model diagram with UML, sequence diagrams, creating gen/spec diagrams, use case scenarios, and activity diagrams. Through several examples and Consulting Opportunities, this chapter demonstrates how to use an object-oriented approach. Consulting Opportunities, diagrams, and problems enable students to learn and use UML to model systems from an object-oriented perspective. Students learn the appropriate situations for using an object-oriented approach. This chapter helps students to decide whether to use the SDLC, the agile approach, or object-oriented systems analysis and design to develop a system.

Part IV (Chapters 11–14) covers the essentials of design. It begins with designing output, because many practitioners believe systems to be output driven. The design of Web-based forms is covered in detail. Particular attention is paid to relating output method to content, the effect of output on users, and designing good forms and screens. Chapter 11 compares advantages and disadvantages of output, including Web displays, audio, DVD, and electronic output such as email and RSS feeds. Designing a Web site for ecommerce purposes is emphasized, and output production and XML is covered. Chapter 12 includes innovative material on designing Web-based input forms, as well as other electronic forms design. Also included is computer-assisted forms design.

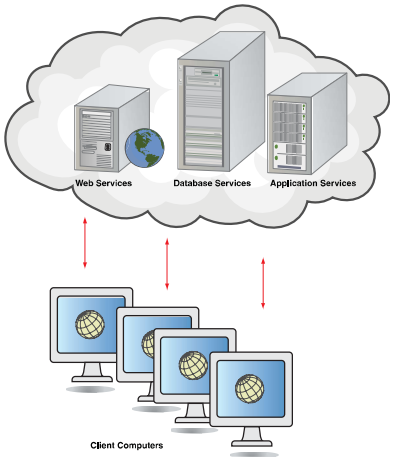
Chapter 12 features in-depth coverage of Web site design, including guidelines on when designers should add video, audio, and animation to Web site designs. The chapter also covers uses of Web push and pull technologies for output designs. There is detailed consideration of how to create effective graphics for corporate Web sites and ways to design effective onscreen navigation for Web site users.

Coverage of intranet and extranet page design is also included. Consideration of database integrity constraints has been included as well, in addition to how the user interacts with the computer and how to design an appropriate interface. The importance of user feedback is also found in Part IV. How to design accurate data entry procedures that take full advantage of computer and human capabilities to assure entry of quality data is emphasized here.

Chapter 13 demonstrates how to use the entity-relationship diagram to determine record keys, as well as providing guidelines for file/database relation design. Students are shown the relevance of database design for the overall usefulness of the system, and how users actually use databases.

Chapter 14 emphasizes Human–Computer Interaction (HCI), especially as it relates to interface design. It introduces HCI, discussing its importance in designing systems that suit individuals and assisting them in achieving personal and organizational goals through their use of information technology. The concepts of usability, fit, perceived usefulness, and perceived ease of use are introduced, as is the Technology Acceptance Model (TAM), so that systems students can knowledgeably incorporate HCI practices into their designs. Chapter 14 also features material on designing easy onscreen navigation for Web site visitors. The chapter presents innovative approaches to searching on the Web, highlights material on GUI design, and provides innovative approaches to designing dialogs. Chapter 14 articulates specialized design considerations for ecommerce Web sites. Mashups, new applications created by combining two or more Web-based application programming interfaces, are also introduced. Chapter 14 also includes extensive coverage on how to formulate queries, all within the framework of HCI.

Part V (Chapters 15 and 16) concludes the book. Chapter 15 focuses on designing accurate data entry procedures and includes material on managing the supply chain through the effective design of business-to-business (B2B) ecommerce. Chapter 16 emphasizes taking a total quality approach to improving software design and maintenance. In addition, material on system security and firewalls is included. Testing, auditing, and maintenance of systems are discussed in the context of total quality management. This chapter helps students understand how service-oriented architecture (SOA) and cloud computing are changing the nature of information systems design. In addition, students learn how to design appropriate training programs for users of the new system, how to recognize the differences among physical conversion strategies, and how to be able to recommend an appropriate one to a client. Chapter 16 also presents techniques for modeling networks, which can be done with popular tools such as Microsoft Visio.



Material on security and privacy in relation to designing ecommerce applications is included. Coverage of security, specifically firewalls, gateways, public key infrastructure (PKI), secure electronic translation (SET), secure socket layering (SSL), virus protection software, URL filtering products, email filtering products, and virtual private networks (VPN), is included. Additionally, current topics of interest to designers of ecommerce applications, including the development and posting of corporate privacy policies, are covered.

Important coverage of how the analyst can promote and then monitor a corporate Web site is included in this section, which features Web activity monitoring, Web site promotion, Web traffic analysis, and audience profiling to ensure the effectiveness of new ecommerce systems. Techniques for evaluating the completed information systems project are covered systematically as well.

The eighth edition contains an updated **Glossary** of terms and a separate list of updated **Acronyms** used in the book and in the systems analysis and design field.

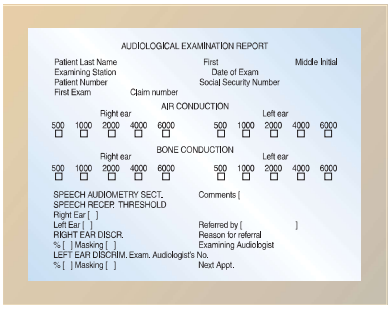
PEDAGOGICAL FEATURES

Chapters in the eighth edition contain:

- **Learning Objectives** at the beginning of each chapter
- **Summaries** that tie together the salient points of each chapter while providing an excellent source of review for exams
- **Keywords and Phrases**
- **Review Questions**
- **Problems**
- **Group Projects** that help students work together in a systems team to solve important problems that are best solved through group interaction
- **Consulting Opportunities**—now more than 60 minicases throughout the book

- **Mac Appeal** columns that update students on design software available on the Mac and iPhone
- **HyperCase Experiences**
- **CPU Case Episodes**—parts of an ongoing case threaded throughout the book

CONSULTING OPPORTUNITIES


AUDIOLOGICAL EXAMINATION REPORT
Patient Last Name First Middle Initial
Examining Station Date of Exam
Patient Number Social Security Number
First Exam Claim number
AIR CONDUCTION
Right ear Left ear
500 1000 2000 4000 6000 500 1000 2000 4000 6000
BONE CONDUCTION
Right ear Left ear
500 1000 2000 4000 6000 500 1000 2000 4000 6000
SPEECH AUDIOMETRY SECT. Comments [
SPEECH RECEP. THRESHOLD
Right Ear [ ]
Left Ear [ ] Referred by [ ]
RIGHT EAR DISCR. Reason for referral
% [ ] Masking [ ] Examining Audiologist
LEFT EAR DISCRIM. Exam. Audiologist's No.
% [ ] Masking [ ] Next Appt.

The eighth edition presents more than 60 Consulting Opportunities, and many of them address relevant and emerging topics that have arisen in the field, including designing systems from an HCI perspective, ecommerce applications for the Web, COTS software, and using UML to model information systems from an object-oriented perspective. Consulting Opportunities can be used for stimulating in-class discussions, or assigned as homework or take-home exam questions.

Because not all systems are extended two- or three-year projects, our book contains many Consulting Opportunities that can be solved quickly in 20 to 30 minutes of group discussion or individual writing. These minicases, written in a humorous manner to enliven the material, require students to synthesize what they have learned up to that point in the course, ask students to mature in their professional and ethical judgment, and expect students to articulate the reasoning that led to their systems decisions.

HYPERCASE EXPERIENCES

HYPERCASE® EXPERIENCE 3.2

"Sometimes the people who have been here for some time are surprised at how much we have actually grown. Yes, I do admit that it isn't easy to keep track of what each person is up to or even what purchases each department has made in the way of hardware and software. We're working on it, though. Snowden would like to see more accountability for computer purchases. He wants to make sure we know what we have, where it is, why we have it, who's using it, and if it's boosting MRE productivity, or, as he so delicately puts it, 'to see whether it's just an expensive toy' that we can live without."

HYPERCASE Questions

1. Compile a computer equipment inventory for the Training and Management Systems Unit, describing all the systems you find. Hint: Create an inventory form to simplify your task.
2. Using the software evaluation guidelines given in the text, do a brief evaluation of GEMS, a software package used by the Management Systems employees. In a paragraph, briefly critique this custom software by comparing it with commercial off-the-shelf software, such as Microsoft Project.
3. List the intangible costs and benefits of GEMS as reported by employees of MRE.
4. Briefly describe the two alternatives Snowden is considering for the proposed project tracking and reporting system.
5. What organizational and political factors should Snowden consider in proposing his new system at MRE? (In a brief paragraph, discuss three central conflicts.)

FIGURE 3.HC1
The reception room resembles a typical corporation. While you are in this HyperCase screen, find the directory if you want to visit someone.

HyperCase® Experiences that pose challenging student exercises are present in each chapter. New scenarios, graphics, and problems to accompany HyperCase version 2.8 are included in the eighth edition. HyperCase has organizational problems featuring state-of-the-art technological systems. HyperCase represents an original virtual organization that allows students who access it to become immediately immersed in organizational life. Students will interview people, observe office environments, analyze their prototypes, and review the documentation of their existing systems. HyperCase 2.8 is Web-based, interactive software that presents an organization called Maple Ridge Engineering (MRE) in a colorful, three-dimensional graphics environment. HyperCase permits professors to begin approaching the systems analysis and design class with exciting multimedia material. Carefully watching their use of time and managing multiple methods, students use the hypertext characteristics of HyperCase on the Web to create their own individual paths through the organization.

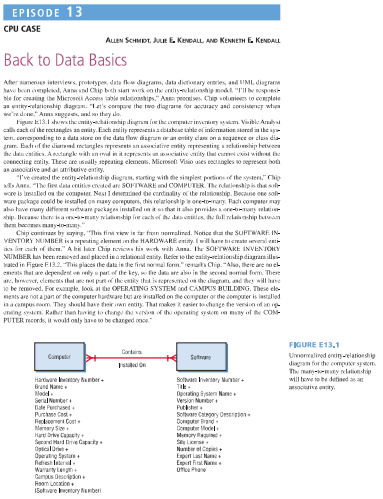
Maple Ridge Engineering is drawn from the actual consulting experiences of the authors of the original version (Raymond Barnes, Richard Baskerville, Julie E. Kendall, and Kenneth E. Kendall). Allen Schmidt joined the project for version 2.0 and has remained with it. Peter Schmidt was the HTML programmer, and Jason Reed created the images for the initial Web version.

In each chapter, there are newly updated HyperCase Experiences that include assignments (and even some clues) to help students solve the difficult organizational problems including developing new systems, merging departments, hiring of employees, security, ecommerce, and disaster recovery planning they encounter at MRE. HyperCase has been fully tested in classrooms, and was an award winner in the Decision Sciences Institute Innovative Instruction competition.

CPU CASE EPISODES

In keeping with our belief that a variety of approaches are important, we have once again integrated the Central Pacific University (CPU) case into every chapter of the eighth edition. The

CPU case makes use of Microsoft Access, Microsoft Visio, and the popular CASE tool Visible Analyst by Visible Systems, Inc., for the example screen shots and the student exercises.


EPISODE 13

CPU CASE

ALLEN SCHMIDT, JULIE E. KENDALL, AND KENNETH E. KENDALL

Back to Data Basics

After numerous interviews, prototypes, data flow diagrams, data dictionary entries, and UML diagrams have been completed, Anna and Chip both start work on the entity-relationship model. "I'll be responsible for creating the Microsoft Access table relationships," Anna promises. Chip volunteers to complete an entity-relationship diagram. "Let's compare the two diagrams for accuracy and consistency when we're done," Anna suggests, and so they do.

Figure E13.1 shows the entity-relationship diagram for the computer inventory system. Visible Analyst calls each of the rectangles an entity. Each entity represents a database table or information stored in the system, corresponding to a data store on the data flow diagram or an entity class on a sequence or class diagram. Each of the diamond rectangles represents an associative entity representing a relationship between the data entities. A rectangle with an oval in it represents an associative entity that cannot exist without the connecting entity. These are usually repeating elements. Microsoft Visio uses rectangles to represent both an associative and an attributive entity.

"I've created the entity-relationship diagram, starting with the simplest portions of the system," Chip tells Anna. "The first data entities created are SOFTWARE and COMPUTER. The relationship is that software is installed on the computer. Step 1 determined the cardinality of the relationship. Because one software package could be installed on many computers, this relationship is one-to-many. Each computer may also have many different software packages installed on it so that it also provides a one-to-many relationship. Because there is a one-to-many relationship for each of the data entities, the full relationship between them becomes many-to-many.

Chip continues by saying, "This first view is far from normalized. Notice that the SOFTWARE INVENTORY NUMBER is a repeating element on the HARDWARE entity. I will have to create several entities for each of them." A bit later Chip resolves his work with Anna. The SOFTWARE INVENTORY NUMBER has been removed and placed in a relational entity. Refer to the entity-relationship diagram illustrated in Figure E13.2. "This places the data in the first normal form," remarks Chip. "Also, there are no elements that are dependent on only a part of the key, so the data are also in the second normal form. There are, however, elements that are not part of the entity that is represented on the diagram, and they will have to be removed. For example, look at the OPERATING SYSTEM and CAMPUS BUILDING. These elements are not a part of the computer hardware but are installed on the computer or the computer is installed in a campus room. They should have their own entity. That makes it easier to change the version of an operating system. Rather than having to change the version of the operating system on many of the COMPUTER records, it would only have to be changed once."

Computer — Contains / Installed On — Software

Hardware Inventory Number +
Brand Name +
Model +
Serial Number +
Date Purchased +
Purchase Cost +
Replacement Cost +
Memory Size +
Hard Drive Capacity +
Second Hard Drive Capacity +
Optical Drive +
Operating System +
Refresh Interval +
Warranty Length +
Campus Description +
Room Location +
(Software Inventory Number)

Software Inventory Number +
Title +
Operating System Name +
Version Number +
Publisher +
Software Category Description +
Computer Brand +
Computer Model +
Memory Required +
Site License +
Number of Copies +
Expert Last Name +
Expert First Name +
Office Phone

FIGURE E13.1
Unnormalized entity-relationship diagram for the computer system. The many-to-many relationship will have to be defined as an associative entity.

The CPU case takes students through all phases of the systems development life cycle. This running case gives students an opportunity to solve problems on their own, using a variety of tools and data that users of the book can download from the Web containing Microsoft Visio, Microsoft Access, and Visible Analyst exercises specifically keyed to each chapter of the book. Additionally, partially completed exercises in Microsoft Access files are available for student use on the Web. The CPU case has been fully tested in classrooms around the world with a variety of students over numerous terms. The case is detailed, rigorous, and rich enough to stand alone as a systems analysis and design project spanning one or two terms. Alternatively, the CPU case can be used as a way to teach the use of CASE tools in conjunction with the assignment of a one- or two-term, real-world project outside the classroom.

EXPANDED WEB SUPPORT

Kendall & Kendall's *Systems Analysis and Design,* Eighth Edition, features Web-based support for solid but lively pedagogical techniques in the information systems field.

- The Web site, located at **www.pearsonhighered.com/kendall,** contains a wealth of critical learning and support tools, which keep class discussions exciting.
- **HyperCase 2.8,** is an award-winning, virtually interactive organization game. Students are encouraged to interview people in the organization, analyze problems, modify data flow diagrams and data dictionaries, react to prototypes, and design new input and output. HyperCase now has a distinctive 3-D look.
- **Student Exercises based on the ongoing CPU case,** with partially solved problems and examples stored in Microsoft Access and Visible Analyst files, allow students to develop a Web-based computer management system.

EXPANDED INSTRUCTOR SUPPLEMENTAL WEB SUPPORT

Extended support for instructors using this edition can be found at the official Web site located at www.pearsonhighered.com/kendall. Resources include:

- A complete set of **PowerPoint presentation slides** for use in lectures
- **Image Library,** a collection of all text art organized by chapter
- **Instructor's Manual** with answers to problems, solutions to cases, and suggestions for approaching the subject matter
- **Test Item File** in Microsoft Word and TestGen with WebCT- and Blackboard-ready conversions
- **Solutions to Student Exercises** based on the ongoing CPU case, with solutions and examples stored in Visible Analyst files and Microsoft Access files.

This page intentionally left blank

ACKNOWLEDGMENTS

Rapid and dramatic changes in IT occurred as we were writing the eighth edition of *Systems Analysis and Design*. We are delighted that this edition is being published at the right time for us to capture many of these changes in systems development.

One major change is that three primary approaches to development are emerging the SDLC, agile approaches, and object-oriented systems analysis and design—and we are able to show where and in what situations each is useful to you as a systems analyst.

Another big change is the rapidly increasing use of the Web as a platform for information systems. Service-oriented architecture and cloud computing both change the way that the analyst approaches designing systems solutions. Along with the Web, analysts are being pushed to design for a wide spectrum of emerging information technologies such as wireless and mobile technologies, enterprise systems, and virtual contexts such as virtual teams and virtual organizations.

Another major change addressed in this edition is the ability of users to personalize and customize their desktops, workspaces, and Web pages, and even for users to alter the professional designs of systems analysts. Analysts see the big picture that users cannot see, and they must always be aware of the organizational impacts of changing systems.

Throughout the book you will learn and apply numerous techniques, methods, tools, and approaches. But when the time comes to interpret what is happening in the organization and to develop meaningful information systems from the application of rules to your analysis, your training combines with creativity to produce a system that is in some ways a surprise: It is structured, yet intuitive; multilayered and complex, in keeping with the character of the organization and uniquely reflective of you as a systems analyst and a human being.

Our students deserve credit for this new edition by providing feedback and suggestions for improvements and asking for increased depth in certain topics. Students told us that they rapidly put to use the new material on object-oriented systems analysis and design as well as that on agile modeling. Their eagerness to teach us new things keeps the book fresh. We want to thank our coauthor, Allen Schmidt, who once again worked with us on the *CPU Case Episodes* and HyperCase 2.8, for all of his hard work, dedication, and humor during our collaboration. He is a wonderful person. Our appreciation also goes to Peter Schmidt and Jason Reed for their improvements to the early HyperCase. We also want to thank the other two original authors of HyperCase, Richard Baskerville and Raymond Barnes, who contributed so much.

We would like to thank our eighth edition production team, especially our executive editor, Bob Horan, whose wisdom and calm demeanor are always inspiring. We are also grateful to Kelly Loftus, our extremely capable assistant editor, for her unruffled competency and for her optimism in keeping the project going. Ana Jankowski, our production editor, also deserves thanks for helping us succeed in making this a strong, complete, and accurate revision. Their help and enthusiasm facilitated the completion of the project in a smooth and timely manner.

Julie and Ken Kendall personally thank Shrek (Brian d'Arcy James) and all of our dear friends in the theatre and the performing arts.

We are also grateful for the encouragement and support of the entire Rutgers community, including our Chancellor Wendell Prittchett, our colleagues in the School of Business-Camden and throughout all of Rutgers, our staff, and our Board of Governors. They have been very enthusiastic about this edition as well as the many translations of *Systems Analysis and Design* available in Spanish, Chinese, and Indonesian.

All the reviewers for the eighth edition deserve our thanks as well. Their thoughtful comments and suggestions helped to strengthen the book. They are:

Stephen T. Brower, Raritan Valley Community College
Robert F. Cope III, Southeastern Louisiana University
Junhua Ding, East Carolina University
Jon Gant, University of Illinois
Cliff Layton, Rogers State University
Keng Siau, University of Nebraska–Lincoln

Many of our colleagues and friends have encouraged us through the process of writing this book. We thank them for their comments on our work. They include: Ayman Abu Hamdieh; Macedonio Alanis; Michel Avital; the Ciupeks; Charles J. Coleman; Roger T. Danforth; Gordon Davis; EgoPo; Paul Gray; Nancy V. Gulick; Andy and Pam Hamingson; Blake Ives; Richard Kalina; Carol Latta; Ken and Jane Laudon; Richard Levao; Joel and Bobbie Porter; Caryn Schmidt; Marc and Jill Schniederjans; Gabriel Shanks; Detmar W. Straub, Jr.; the Vargos; Merrill Warkentin; Jeff and Bonnie Weil; Ping Zhang, and all of our friends and colleagues in the Association for Information Systems, the Decision Sciences Institute, IFIP Working Group 8.2, and all those involved in the PhD Project (founded by the KPMG Foundation), which serves minority doctoral students in information systems.

Our heartfelt thanks go to the memory of Julia A. Kendall and to the memory of Edward J. Kendall. Their belief that love, goals, and hard work are an unbeatable combination continues to infuse our every endeavor.

PART I
Systems Analysis Fundamentals

CHAPTER 1

Systems, Roles, and Development Methodologies

LEARNING OBJECTIVES

Once you have mastered the material in this chapter you will be able to:

1. **Recall the basic types of computer-based systems that a systems analyst needs to address.**
2. **Understand how users working in context with new technologies change the dynamics of a system.**
3. **Realize what the many roles of a systems analyst are.**
4. **Comprehend the fundamentals of three development methodologies: SDLC, the agile approach, and object-oriented systems analysis and design.**
5. **Understand what CASE tools are and how they help a systems analyst.**

Organizations have long recognized the importance of managing key resources such as people and raw materials. Information has now moved to its rightful place as a key resource. Decision makers now understand that information is not just a by-product of conducting business; rather, it fuels business and can be the critical factor in determining the success or failure of a business.

To maximize the usefulness of information, a business must manage it correctly, just as it manages other resources. Managers need to understand that costs are associated with the production, distribution, security, storage, and retrieval of all information. Although information is all around us, it is not free, and its strategic use for positioning a business competitively should not be taken for granted.

The ready availability of networked computers, along with access to the Internet and the Web, has created an information explosion throughout society in general and business in particular. Managing computer-generated information differs in significant ways from handling manually produced data. Usually there is a greater quantity of computer information to administer. Costs of organizing and maintaining it can increase at alarming rates, and users often treat it less skeptically than information obtained in different ways. This chapter examines the fundamentals of different kinds of information systems, the varied roles of systems analysts, and the phases in the systems development life cycle (SDLC) as they relate to Human–Computer Interaction (HCI) factors; it also introduces Computer-Aided Software Engineering (CASE) tools.

TYPES OF SYSTEMS

Information systems are developed for different purposes, depending on the needs of human users and the business. Transaction processing systems (TPS) function at the operational level of the organization; office automation systems (OAS) and knowledge work systems (KWS) support work at the knowledge level. Higher-level systems include management information systems (MIS) and decision support systems (DSS). Expert systems apply the expertise of decision makers to solve specific, structured problems. On the strategic level of management we find executive support systems (ESS). Group decision support systems (GDSS) and the more generally described computer-supported collaborative work systems (CSCWS) aid group-level decision making of a semistructured or unstructured variety.

The variety of information systems that analysts may develop is shown in Figure 1.1. Notice that the figure presents these systems from the bottom up, indicating that the operational, or lowest, level of the organization is supported by TPS, and the strategic, or highest, level of semistructured and unstructured decisions is supported by ESS, GDSS, and CSCWS at the top. This text uses the terms *management information systems, information systems* (IS), *computerized information systems,* and *computerized business information systems* interchangeably to denote computerized information systems that support the broadest range of user interactions with technologies and business activities through the information they produce in organizational contexts.

Transaction Processing Systems

Transaction processing systems (TPS) are computerized information systems that were developed to process large amounts of data for routine business transactions such as payroll and inventory. A TPS eliminates the tedium of necessary operational transactions and reduces the time once required to perform them manually, although people must still input data to computerized systems.

Transaction processing systems are boundary-spanning systems that permit the organization to interact with external environments. Because managers look to the data generated by the TPS for up-to-the-minute information about what is happening in their companies, it is essential to the day-to-day operations of business that these systems function smoothly and without interruption.

Office Automation Systems and Knowledge Work Systems

At the knowledge level of the organization are two classes of systems. Office automation systems (OAS) support data workers, who do not usually create new knowledge but rather analyze information to transform data or manipulate it in some way before sharing it with, or formally disseminating it throughout, the organization and, sometimes, beyond. Familiar aspects of OAS include

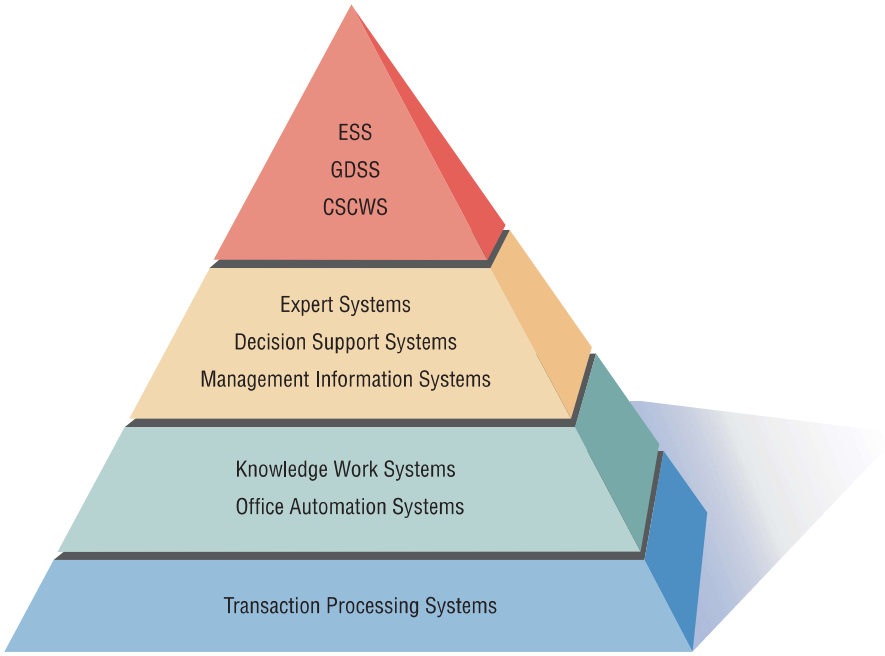


FIGURE 1.1

A systems analyst may be involved with any or all of these systems.

word processing, spreadsheets, desktop publishing, electronic scheduling, and communication through voice mail, email (electronic mail), and teleconferencing.

Knowledge work systems (KWS) support professional workers such as scientists, engineers, and doctors by aiding them in their efforts to create new knowledge (often in teams) and by allowing them to contribute it to their organization or to society at large.

Management Information Systems

Management information systems (MIS) do not replace transaction processing systems; rather, all MIS include transaction processing. MIS are computerized information systems that work because of the purposeful interaction between people and computers. By requiring people, software, and hardware to function in concert, management information systems support users in accomplishing a broader spectrum of organizational tasks than transaction processing systems, including decision analysis and decision making.

To access information, users of the management information system share a common database. The database stores both data and models that help the user interact with, interpret, and apply that data. Management information systems output information that is used in decision making. A management information system can also help integrate some of the computerized information functions of a business.

Decision Support Systems

A higher-level class of computerized information systems is decision support systems (DSS). DSS are similar to the traditional management information system because they both depend on a database as a source of data. A decision support system departs from the traditional management information system because it emphasizes the support of decision making in all its phases, although the actual decision is still the exclusive province of the decision maker. Decision support systems are more closely tailored to the person or group using them than is a traditional management information system. Sometimes they are discussed as systems that focus on business intelligence.

Artificial Intelligence and Expert Systems

Artificial intelligence (AI) can be considered the overarching field for expert systems. The general thrust of AI has been to develop machines that behave intelligently. Two avenues of AI research are (1) understanding natural language and (2) analyzing the ability to reason through a problem to its logical conclusion. Expert systems use the approaches of AI reasoning to solve the problems put to them by business (and other) users.

Expert systems are a very special class of information system that has been made practicable for use by business as a result of widespread availability of hardware and software such as personal computers (PCs) and expert system shells. An expert system (also called a knowledge-based system) effectively captures and uses the knowledge of a human expert or experts for solving a particular problem experienced in an organization. Notice that unlike DSS, which leave the ultimate judgment to the decision maker, an expert system selects the best solution to a problem or a specific class of problems.

The basic components of an expert system are the knowledge base, an inference engine connecting the user with the system by processing queries via languages such as structured query language (SQL), and the user interface. People called knowledge engineers capture the expertise of experts, build a computer system that includes this expert knowledge, and then implement it.

Group Decision Support Systems and Computer-Supported Collaborative Work Systems

Organizations are becoming increasingly reliant on groups or teams to make decisions together. When groups make semistructured or unstructured decisions, a group decision support system may afford a solution. Group decision support systems (GDSS), which are used in special rooms equipped in a number of different configurations, permit group members to interact with electronic support—often in the form of specialized software—and a special group facilitator. Group decision support systems are intended to bring a group together to solve a problem with the help of various supports such as polling, questionnaires, brainstorming, and scenario creation. GDSS software can be designed to minimize typical negative group behaviors such as lack of participation due to fear

of reprisal for expressing an unpopular or contested viewpoint, domination by vocal group members, and "group think" decision making. Sometimes GDSS are discussed under the more general term *computer-supported collaborative work systems* (CSCWS), which might include software support called groupware for team collaboration via networked computers. Group decision support systems can also be used in a virtual setting.

Executive Support Systems

When executives turn to the computer, they are often looking for ways to help them make decisions on the strategic level. Executive support systems (ESS) help executives organize their interactions with the external environment by providing graphics and communications technologies in accessible places such as boardrooms or personal corporate offices. Although ESS rely on the information generated by TPS and MIS, executive support systems help their users address unstructured decision problems, which are not application specific, by creating an environment that helps them think about strategic problems in an informed way. ESS extend and support the capabilities of executives, permitting them to make sense of their environments.

INTEGRATING TECHNOLOGIES FOR SYSTEMS

As users adopt new technologies, some of the systems analyst's work will be devoted to integrating traditional systems with new ones to ensure a useful context, as shown in Figure 1.2. This section describes some of the new information technologies systems analysts will be using as people work to integrate their ecommerce applications into their traditional businesses or as they begin entirely new ebusinesses.

Ecommerce Applications and Web Systems

Many of the systems discussed here can be imbued with greater functionality if they are migrated to the World Wide Web or if they are originally conceived and implemented as Web-based technologies. There are many benefits to mounting or improving an application on the Web:

1. Increasing user awareness of the availability of a service, product, industry, person, or group.
2. The possibility of 24-hour access for users.

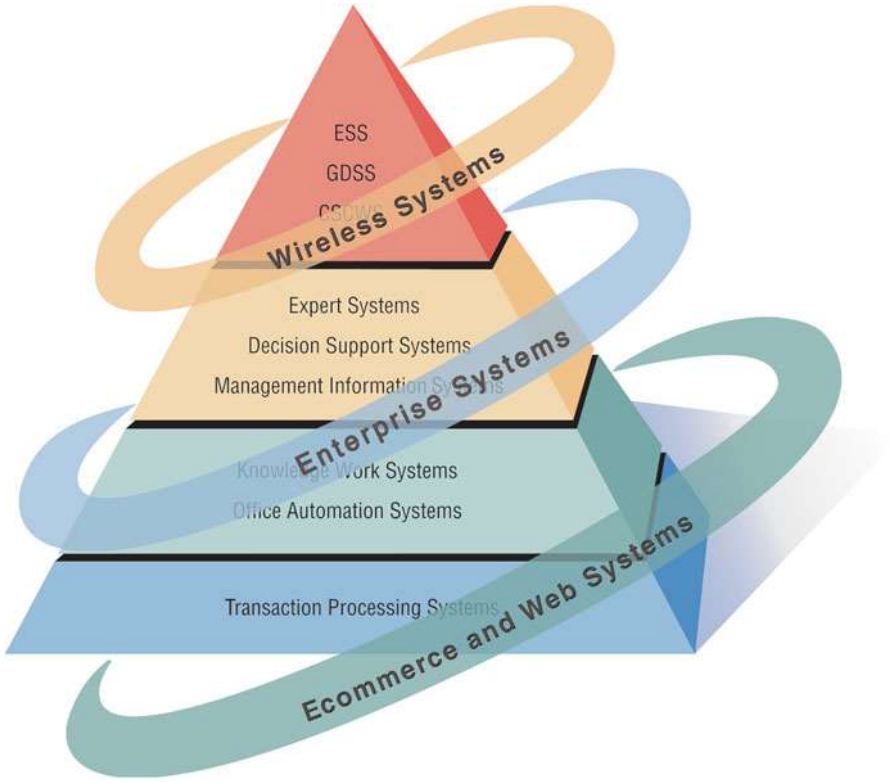


FIGURE 1.2

Systems analysts need to be aware that integrating technologies affect all types of users and systems.

3. Improving the usefulness and usability of the interface design.
4. Creating a system that can extend globally rather than remain local, thus reaching people in remote locations without worry of the time zone in which they are located.

Enterprise Systems

Many organizations envision potential benefits from the integration of many information systems existing on different management levels and within different functions. Some authors discuss integration as service-oriented architecture (SOA), which exists in layers. Enterprise systems would comprise the top layer. Enterprise systems, also called enterprise resource planning (ERP) systems, are designed to perform this integration. Instituting ERP requires enormous commitment and organizational change. Often systems analysts serve as consultants to ERP endeavors that use proprietary software. Popular ERP software includes that from SAP and Oracle. Some of these packages are targeted toward moving enterprises onto the Web. Typically, analysts as well as some users require vendor training, support, and maintenance to be able to properly design, install, maintain, update, and use a particular ERP package.

Systems for Wireless and Mobile Devices

Analysts are being asked to design a plethora of new systems and applications for adventurous users, including many for wireless and mobile devices such as the Apple iPhone, iPod, or the BlackBerry. In addition, analysts may find themselves designing standard or wireless communications networks for users that integrate voice, video, text messaging, and email into organizational intranets or industry extranets. Wireless ecommerce is referred to as mcommerce (mobile commerce).

Wireless local area networks (WLANs); wireless fidelity networks, called Wi-Fi; and personal wireless networks that bring together many types of devices under the standard called Bluetooth are all systems that you may be asked to design. In more advanced settings, analysts may be called on to design intelligent agents, software that can assist users with tasks in which the software learns users' preferences over time and then acts on those preferences. For example, in the use of pull technology, an intelligent agent would search the Web for stories of interest to the user, having observed the user's behavior patterns with information over time, and would conduct searches on the Web without continual prompting from the user.

Open Source Software

An alternative to traditional software development in which proprietary code is hidden from the users is called open source software (OSS). With OSS, the code, or computer instructions, can be studied, shared, and modified by many users and programmers. Rules of this community include the idea that any program modifications must be shared with all the people on the project.

Development of OSS has also been characterized as a philosophy rather than simply as the process of creating new software. Often those involved in OSS communities view it as a way to help societies change. Widely known open source projects include Apache for developing a Web server, the browser called Mozilla Firefox, and Linux, which is a Unix-like open source operating system.

However, it would be an oversimplification to think of OSS as a monolithic movement, and it does little to reveal what type of users or user analysts are developing OSS projects and on what basis. To help us understand the open source movement, researchers have recently categorized open source communities into four community types—ad hoc, standardized, organized, and commercial—along six different dimensions—general structure, environment, goals, methods, user community, and licensing. Some researchers argue that OSS is at a crossroads and that the commercial and community OSS groups need to understand where they converge and where the potential for conflict exists.

Open source development is useful for many applications running on diverse technologies, including handheld devices and communication equipment. Its use may encourage progress in creating standards for devices to communicate more easily. Widespread use of OSS may alleviate some of the severe shortages of programmers by placing programming tools in the hands of students in developing countries sooner than if they were limited to using proprietary packages, and it may lead to solving large problems through intense and extensive collaboration.

NEED FOR SYSTEMS ANALYSIS AND DESIGN

Systems analysis and design, as performed by systems analysts, seeks to understand what humans need to analyze data input or data flow systematically, process or transform data, store data, and output information in the context of a particular organization or enterprise. By doing thorough analysis, analysts seek to identify and solve the right problems. Furthermore, systems analysis and design is used to analyze, design, and implement improvements in the support of users and the functioning of businesses that can be accomplished through the use of computerized information systems.

Installing a system without proper planning leads to great user dissatisfaction and frequently causes the system to fall into disuse. Systems analysis and design lends structure to the analysis and design of information systems, a costly endeavor that might otherwise have been done in a haphazard way. It can be thought of as a series of processes systematically undertaken to improve a business through the use of computerized information systems. Systems analysis and design involves working with current and eventual users of information systems to support them in working with technologies in an organizational setting.

User involvement throughout the systems project is critical to the successful development of computerized information systems. Systems analysts, whose roles in the organization are discussed next, are the other essential component in developing useful information systems.

Users are moving to the forefront as software development teams become more international in their composition. This means that there is more emphasis on working with software users; on performing analysis of their business, problems, and objectives; and on communicating the analysis and design of the planned system to all involved.

New technologies also are driving the need for systems analysis. Ajax (Asynchronous JavaScript and XML) is not a new programming language, but a technique that uses existing languages to make Web pages function more like a traditional desktop application program. Building and redesigning Web pages that utilize Ajax technologies will be a task facing analysts. New programming languages, such as the open source Web framework, *Ruby on Rails*, which is a combination programming language and code generator for creating Web applications, will require more analysis.

ROLES OF THE SYSTEMS ANALYST

The systems analyst systematically assesses how users interact with technology and how businesses function by examining the inputting and processing of data and the outputting of information with the intent of improving organizational processes. Many improvements involve better support of users' work tasks and business functions through the use of computerized information systems. This definition emphasizes a systematic, methodical approach to analyzing—and potentially improving—what is occurring in the specific context experienced by users and created by a business.

Our definition of a systems analyst is necessarily broad. The analyst must be able to work with people of all descriptions and be experienced in working with computers. The analyst plays many roles, sometimes balancing several at the same time. The three primary roles of the systems analyst are consultant, supporting expert, and agent of change.

Systems Analyst as Consultant

The systems analyst frequently acts as a systems consultant to humans and their businesses and, thus, may be hired specifically to address information systems issues within a business. Such hiring can be an advantage because outside consultants can bring with them a fresh perspective that other people in an organization do not possess. It also means that outside analysts are at a disadvantage because an outsider can never know the true organizational culture. As an outside consultant, you will rely heavily on the systematic methods discussed throughout this text to analyze and design appropriate information systems for users working in a particular business. In addition, you will rely on information systems users to help you understand the organizational culture from others' viewpoints.

Systems Analyst as Supporting Expert

Another role that you may be required to play is that of supporting expert within a business for which you are regularly employed in some systems capacity. In this role the analyst draws on professional expertise concerning computer hardware and software and their uses in the business.

CONSULTING OPPORTUNITY 1.1

Healthy Hiring: Ecommerce Help Wanted

"You'll be happy to know that we made a strong case to management that we should hire a new systems analyst to specialize in ecommerce development," says Al Falfa, a systems analyst for the multioutlet international chain of Marathon Vitamin Shops. He is meeting with his large team of systems analysts to decide on the qualifications that their new team member should possess. Al continues, saying, "In fact, they were so excited by the possibility of our team helping to move Marathon into an ecommerce strategy that they've said we should start our search now and not wait until the fall."

Ginger Rute, another analyst, agrees, saying, "The demand for Web site developers is still outstripping the supply. We should move quickly. I think our new person should be knowledgeable in system modeling, JavaScript, C++, Rational Rose, and familiar with Ajax, just to name a few."

Al looks surprised at Ginger's long list of skills but then replies, "Well, that's certainly one way we could go. But I would also like to see a person with some business savvy. Most of the people coming out of school will have solid programming skills, but they should know about accounting, inventory, and distribution of goods and services, too."

The newest member of the systems analysis group, Vita Ming, finally breaks into the discussion. She says, "One of the reasons I chose to come to work with all of you was that I thought we all got along quite well together. Because I had some other opportunities, I looked very carefully at what the atmosphere was here. From what I've seen, we're a friendly group. Let's be sure to hire someone who has a good personality and who fits in well with us."

Al concurs, continuing, "Vita's right. The new person should be able to communicate well with us, and with business clients, too. We are always communicating in some way, through formal presentations, drawing diagrams, or interviewing users. If they understand decision making, it will make their job easier, too. Also, Marathon is interested in integrating ecommerce into the entire business. We need someone who at least grasps the strategic importance of the Web. Page design is such a small part of it."

Ginger interjects again with a healthy dose of practicality, saying, "Leave that to management. I still say the new person should be a good programmer." Then she ponders aloud, "I wonder how important UML will be?"

After listening patiently to everyone's wish list, one of the senior analysts, Cal Siem, speaks up, joking, "We'd better see if Superman is available!"

As the group shares a laugh, Al sees an opportunity to try for some consensus, saying, "We've had a chance to hear a number of different qualifications. Let's each take a moment and make a list of the qualifications we personally think are essential for the new ecommerce development person to possess. We'll share them and continue discussing until we can describe the person in enough detail to turn a description over to the human resources group for processing."

What qualifications should the systems analysis team be looking for when hiring their new ecommerce development team member? Is it more important to know specific languages or to have an aptitude for picking up languages and software packages quickly? How important is it that the person being hired has some basic business understanding? Should all team members possess identical competencies and skills? What personality or character traits are desirable in a systems analyst who will be working in ecommerce development?

This work is often not a full-blown systems project, but rather it entails a small modification or decision affecting a single department.

As the supporting expert, you are not managing the project; you are merely serving as a resource for those who are. If you are a systems analyst employed by a manufacturing or service organization, many of your daily activities may be encompassed by this role.

Systems Analyst as Agent of Change

The most comprehensive and responsible role that the systems analyst takes on is that of an agent of change, whether internal or external to the business. As an analyst, you are an agent of change whenever you perform any of the activities in the systems development life cycle (discussed in the next section) and are present and interacting with users and the business for an extended period (from two weeks to more than a year). An agent of change can be defined as a person who serves as a catalyst for change, develops a plan for change, and works with others in facilitating that change.

Your presence in the business changes it. As a systems analyst, you must recognize this fact and use it as a starting point for your analysis. Hence, you must interact with users and management (if they are not one and the same) from the very beginning of your project. Without their help you cannot understand what they need to support their work in the organization, and real change cannot take place.

If change (that is, improvements to the business that can be realized through information systems) seems warranted after analysis, the next step is to develop a plan for change along with the people who must enact the change. Once a consensus is reached on the change that is to be made, you must constantly interact with those who are changing.

As a systems analyst acting as an agent of change, you advocate a particular avenue of change involving the use of information systems. You also teach users the process of change, because changes in the information system do not occur independently; rather, they cause changes in the rest of the organization as well.

Qualities of the Systems Analyst

From the foregoing descriptions of the roles the systems analyst plays, it is easy to see that the successful systems analyst must possess a wide range of qualities. Many different kinds of people are systems analysts, so any description is destined to fall short in some way. There are some qualities, however, that most systems analysts seem to display.

Above all, the analyst is a problem solver. He or she is a person who views the analysis of problems as a challenge and who enjoys devising workable solutions. When necessary, the analyst must be able to systematically tackle the situation at hand through skillful application of tools, techniques, and experience. The analyst must also be a communicator capable of relating meaningfully to other people over extended periods of time. Systems analysts need to be able to understand humans' needs in interacting with technology, and they need enough computer experience to program, to understand the capabilities of computers, to glean information requirements from users, and to communicate what is needed to programmers. They also need to possess strong personal and professional ethics to help them shape their client relationships.

The systems analyst must be a self-disciplined, self-motivated individual who is able to manage and coordinate other people, as well as innumerable project resources. Systems analysis is a demanding career, but, in compensation, an ever-changing and always challenging one.

THE SYSTEMS DEVELOPMENT LIFE CYCLE

Throughout this chapter we have referred to the systematic approach analysts take to the analysis and design of information systems. Much of this is embodied in what is called the systems development life cycle (SDLC). The SDLC is a phased approach to analysis and design that holds that systems are best developed through the use of a specific cycle of analyst and user activities.

Analysts disagree on exactly how many phases there are in the SDLC, but they generally laud its organized approach. Here we have divided the cycle into seven phases, as shown in Figure 1.3. Although each phase is presented discretely, it is never accomplished as a separate step. Instead, several activities can occur simultaneously, and activities may be repeated.

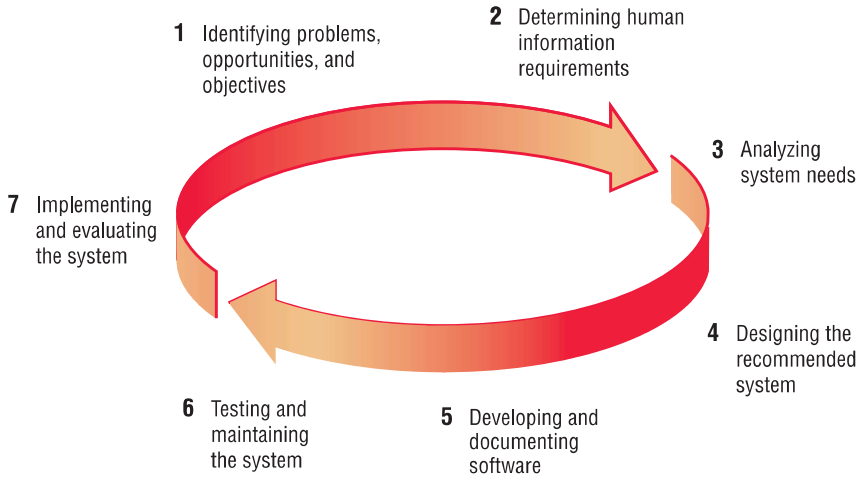


FIGURE 1.3
The seven phases of the systems development life cycle (SDLC).

Incorporating Human–Computer Interaction Considerations

In recent years, the study of human–computer interaction (HCI) has become increasingly important for systems analysts. Although the definition is still evolving, researchers characterize HCI as the "aspect of a computer that enables communications and interactions between humans and the computer. It is the layer of the computer that is between humans and the computer" (Zhang, Carey, Te'eni, & Tremaine, 2005, p. 518). Analysts using an HCI approach are emphasizing people rather than the work to be done or the IT that is involved. Their approach to a problem is multifaceted, looking at the "human ergonomic, cognitive, affective, and behavioral factors involved in user tasks, problem solving processes and interaction context" (Zhang, Carey, Te'eni, & Tremaine, 2005, p. 518). Human computer interaction moves away from focusing first on organizational and system needs and instead concentrates on human needs. Analysts adopting HCI principles examine a variety of user needs in the context of humans interacting with information technology to complete tasks and solve problems. These include taking into account physical or ergonomic factors; usability factors that are often labeled cognitive matters; the pleasing, aesthetic, and enjoyable aspects of using the system; and behavioral aspects that center on the usefulness of the system.

Another way to think about HCI is to think of it as a human-centered approach that puts people ahead of organizational structure or culture when creating new systems. When analysts employ HCI as a lens to filter the world, their work will possess a different quality than the work of those who do not possess this perspective.

Your career can benefit from a strong grounding in HCI. The demand for analysts who are capable of incorporating HCI into the systems development process keeps rising, as companies increasingly realize that the quality of systems and the quality of work life can both be improved by taking a human-centered approach at the outset of a project.

The application of human–computer interaction principles tries to uncover and address the frustrations that users voice over their use of information technology. These concerns include a suspicion that systems analysts misunderstand the work being done, the tasks involved, and how they can best be supported; a feeling of helplessness or lack of control when working with the system; intentional breaches of privacy; trouble navigating through system screens and menus; and a general mismatch between the system designed and the way users themselves think of their work processes.

Misjudgments and errors in design that cause users to neglect new systems or that cause systems to fall into disuse soon after their implementation can be eradicated or minimized when systems analysts adopt an HCI approach.

Researchers in HCI see advantages to the inclusion of HCI in every phase of the SDLC. This is a worthwhile approach, and we will try to mirror this by bringing human concerns explicitly into each phase of the SDLC. As a person who is learning systems analysis, you can also bring a fresh eye to the SDLC to identify opportunities for designers to address HCI concerns and ways for users to become more central to each phase of the SDLC. Chapter 14 is devoted to examining the role of the systems analyst in designing human-centered systems and interfaces from an HCI perspective.

Identifying Problems, Opportunities, and Objectives

In this first phase of the systems development life cycle, the analyst is concerned with correctly identifying problems, opportunities, and objectives. This stage is critical to the success of the rest of the project, because no one wants to waste subsequent time addressing the wrong problem.

The first phase requires that the analyst look honestly at what is occurring in a business. Then, together with other organizational members, the analyst pinpoints problems. Often others will bring up these problems, and they are the reason the analyst was initially called in. Opportunities are situations that the analyst believes can be improved through the use of computerized information systems. Seizing opportunities may allow the business to gain a competitive edge or set an industry standard.

Identifying objectives is also an important component of the first phase. The analyst must first discover what the business is trying to do. Then the analyst will be able to see whether some aspect of information systems applications can help the business reach its objectives by addressing specific problems or opportunities.

The people involved in the first phase are the users, analysts, and systems managers coordinating the project. Activities in this phase consist of interviewing user management, summarizing the

knowledge obtained, estimating the scope of the project, and documenting the results. The output of this phase is a feasibility report containing a problem definition and summarizing the objectives. Management must then make a decision on whether to proceed with the proposed project. If the user group does not have sufficient funds in its budget or wishes to tackle unrelated problems, or if the problems do not require a computer system, a different solution may be recommended, and the systems project does not proceed any further.

Determining Human Information Requirements

The next phase the analyst enters is that of determining the human needs of the users involved, using a variety of tools to understand how users interact in the work context with their current information systems. The analyst will use interactive methods such as interviewing, sampling and investigating hard data, and questionnaires, along with unobtrusive methods, such as observing decision makers' behavior and their office environments, and all-encompassing methods, such as prototyping.

The analyst will use these methods to pose and answer many questions concerning human-computer interaction (HCI), including questions such as, "What are the users' physical strengths and limitations?" In other words, "What needs to be done to make the system audible, legible, and safe?" "How can the new system be designed to be easy to use, learn, and remember?" "How can the system be made pleasing or even fun to use?" "How can the system support a user's individual work tasks and make them more productive in new ways?"

In the information requirements phase of the SDLC, the analyst is striving to understand what information users need to perform their jobs. At this point the analyst is examining how to make the system useful to the people involved. How can the system better support individual tasks that need doing? What new tasks are enabled by the new system that users were unable to do without it? How can the new system be created to extend a user's capabilities beyond what the old system provided? How can the analyst create a system that is rewarding for workers to use?

The people involved in this phase are the analysts and users, typically operations managers and operations workers. The systems analyst needs to know the details of current system functions: the who (the people who are involved), what (the business activity), where (the environment in which the work takes place), when (the timing), and how (how the current procedures are performed) of the business under study. The analyst must then ask why the business uses the current system. There may be good reasons for doing business using the current methods, and these should be considered when designing any new system.

Agile development is an object-oriented approach (OOA) to systems development that includes a method of development (including generating information requirements) as well as software tools. In this text it is paired with prototyping in Chapter 6. (There is more about object-oriented approaches in Chapter 10.)

If the reason for current operations is that "it's always been done that way," however, the analyst may wish to improve on the procedures. At the completion of this phase, the analyst should understand how users accomplish their work when interacting with a computer and begin to know how to make the new system more useful and usable. The analyst should also know how the business functions and have complete information on the people, goals, data, and procedures involved.

Analyzing System Needs

The next phase that the systems analyst undertakes involves analyzing system needs. Again, special tools and techniques help the analyst make requirement determinations. Tools such as data flow diagrams (DFD) to chart the input, processes, and output of the business's functions, or activity diagrams or sequence diagrams to show the sequence of events, illustrate systems in a structured, graphical form. From data flow, sequence, or other diagrams, a data dictionary is developed that lists all the data items used in the system, as well as their specifications.

During this phase the systems analyst also analyzes the structured decisions made. Structured decisions are those for which the conditions, condition alternatives, actions, and action rules can be determined. There are three major methods for analysis of structured decisions: structured English, decision tables, and decision trees.

At this point in the SDLC, the systems analyst prepares a systems proposal that summarizes what has been found out about the users, usability, and usefulness of current systems; provides cost-benefit analyses of alternatives; and makes recommendations on what (if anything) should be done. If one of the recommendations is acceptable to management, the analyst proceeds along

that course. Each systems problem is unique, and there is never just one correct solution. The manner in which a recommendation or solution is formulated depends on the individual qualities and professional training of each analyst and the analyst's interaction with users in the context of their work environment.

Designing the Recommended System

In the design phase of the SDLC, the systems analyst uses the information collected earlier to accomplish the logical design of the information system. The analyst designs procedures for users to help them accurately enter data so that data going into the information system are correct. In addition, the analyst provides for users to complete effective input to the information system by using techniques of good form and Web page or screen design.

Part of the logical design of the information system is devising the HCI. The interface connects the user with the system and is thus extremely important. The user interface is designed with the help of users to make sure that the system is audible, legible, and safe, as well as attractive and enjoyable to use. Examples of physical user interfaces include a keyboard (to type in questions and answers), onscreen menus (to elicit user commands), and a variety of graphical user interfaces (GUIs) that use a mouse or touch screen.

The design phase also includes designing databases that will store much of the data needed by decision makers in the organization. Users benefit from a well-organized database that is logical to them and corresponds to the way they view their work. In this phase the analyst also works with users to design output (either onscreen or printed) that meets their information needs.

Finally, the analyst must design controls and backup procedures to protect the system and the data, and to produce program specification packets for programmers. Each packet should contain input and output layouts, file specifications, and processing details; it may also include decision trees or tables, UML or data flow diagrams, and the names and functions of any prewritten code that is either written in-house or using code or other class libraries.

Developing and Documenting Software

In the fifth phase of the SDLC, the analyst works with programmers to develop any original software that is needed. During this phase the analyst works with users to develop effective documentation for software, including procedure manuals, online help, and Web sites featuring Frequently Asked Questions (FAQs), on Read Me files shipped with new software. Because users are involved from the beginning, phase documentation should address the questions they have raised and solved jointly with the analyst. Documentation tells users how to use software and what to do if software problems occur.

Programmers have a key role in this phase because they design, code, and remove syntactical errors from computer programs. To ensure quality, a programmer may conduct either a design or a code walkthrough, explaining complex portions of the program to a team of other programmers.

Testing and Maintaining the System

Before the information system can be used, it must be tested. It is much less costly to catch problems before the system is signed over to users. Some of the testing is completed by programmers alone, some of it by systems analysts in conjunction with programmers. A series of tests to pinpoint problems is run first with sample data and eventually with actual data from the current system. Often test plans are created early in the SDLC and are refined as the project progresses.

Maintenance of the system and its documentation begins in this phase and is carried out routinely throughout the life of the information system. Much of the programmer's routine work consists of maintenance, and businesses spend a great deal of money on maintenance. Some maintenance, such as program updates, can be done automatically via a vendor site on the Web. Many of the systematic procedures the analyst employs throughout the SDLC can help ensure that maintenance is kept to a minimum.

Implementing and Evaluating the System

In this last phase of systems development, the analyst helps implement the information system. This phase involves training users to handle the system. Vendors do some training, but oversight of training is the responsibility of the systems analyst. In addition, the analyst needs to plan for a smooth conversion from the old system to the new one. This process includes converting files from old formats to new ones, or building a database, installing equipment, and bringing the new system into production.

MAC APPEAL

At home and in our visits to university campuses and businesses around the world, we've noticed that students and organizations are increasingly showing an interest in Macs. Therefore, we thought it would add a little bit of interest to show some Mac options that a systems designer has. At the time we're writing this book, about one out of seven personal computers purchased in the United States is a Mac. Macs are quality Intel-based machines that run under a competent operating system and can also run Windows, so in effect everything that can be done on a PC can be done on a Mac. One way to run Windows is to boot directly into Windows (once it's installed); another is to use virtualization using software such as VM Fusion, which is shown in Figure 1.MAC.

Adopters of Macs have cited many reasons for using Macs including better security built into the Mac operating system, intelligent backup using the built-in time machine, the multitude of applications already included, the reliability of setup and networking, and the ability to sync Macs with other Macs and iPhones. The most compelling reason, we think, is the design itself.

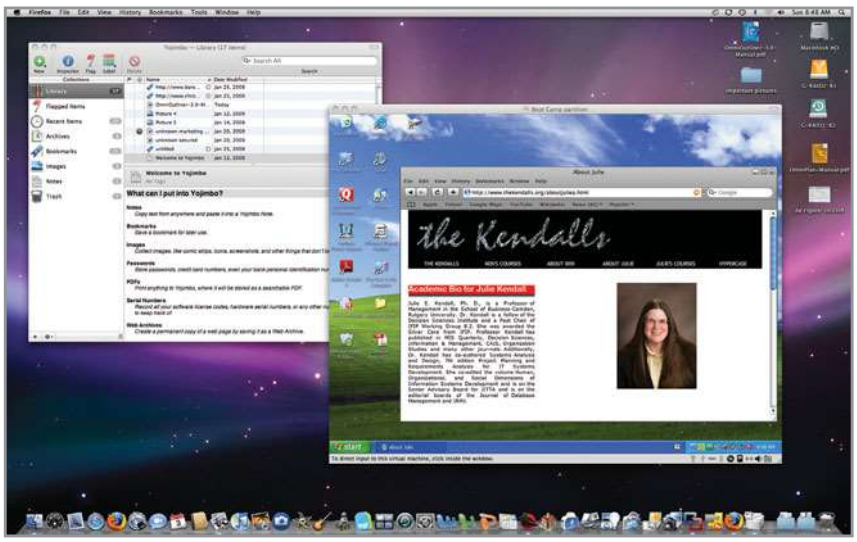


FIGURE 1.MAC

Running Windows on a Mac using Virtualization Software called VM Fusion.

Evaluation is included as part of this final phase of the SDLC mostly for the sake of discussion. Actually, evaluation takes place during every phase. A key criterion that must be satisfied is whether the intended users are indeed using the system.

It should be noted that systems work is often cyclical. When an analyst finishes one phase of systems development and proceeds to the next, the discovery of a problem may force the analyst to return to the previous phase and modify the work done there.

The Impact of Maintenance

After the system is installed, it must be maintained, meaning that the computer programs must be modified and kept up to date. Figure 1.4 illustrates the average amount of time spent on maintenance at a typical MIS installation. Estimates of the time spent by departments on maintenance have ranged from 48 to 60 percent of the total time spent developing systems. Very little time remains for new systems development. As the number of programs written increases, so does the amount of maintenance they require.

Maintenance is performed for two reasons. The first of these is to correct software errors. No matter how thoroughly the system is tested, bugs or errors creep into computer programs. Bugs

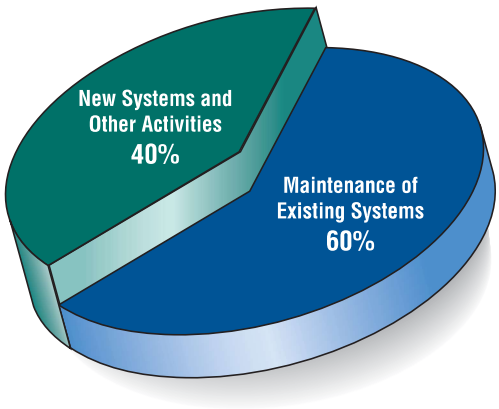


FIGURE 1.4

Some researchers estimate that the amount of time spent on system maintenance may be as much as 60 percent of the total time spent on systems projects.

in commercial PC software are often documented as "known anomalies," and are corrected when new versions of the software are released or in an interim release. In custom software (also called bespoke software), bugs must be corrected as they are detected.

The other reason for performing system maintenance is to enhance the software's capabilities in response to changing organizational needs, generally involving one of the following three situations:

1. Users often request additional features after they become familiar with the computer system and its capabilities.
2. The business changes over time.
3. Hardware and software are changing at an accelerated pace.

Figure 1.5 illustrates the amount of resources—usually time and money—spent on systems development and maintenance. The area under the curve represents the total dollar amount spent. You can see that over time the total cost of maintenance is likely to exceed that of systems development. At a certain point it becomes more feasible to perform a new systems study, because the cost of continued maintenance is clearly greater than that of creating an entirely new information system.

In summary, maintenance is an ongoing process over the life cycle of an information system. After the information system is installed, maintenance usually takes the form of correcting previously undetected program errors. Once these are corrected, the system approaches a steady state, providing dependable service to its users. Maintenance during this period may consist of removing a few previously undetected bugs and updating the system with a few minor enhancements. As time goes on and the business and technology change, however, the maintenance effort increases dramatically.

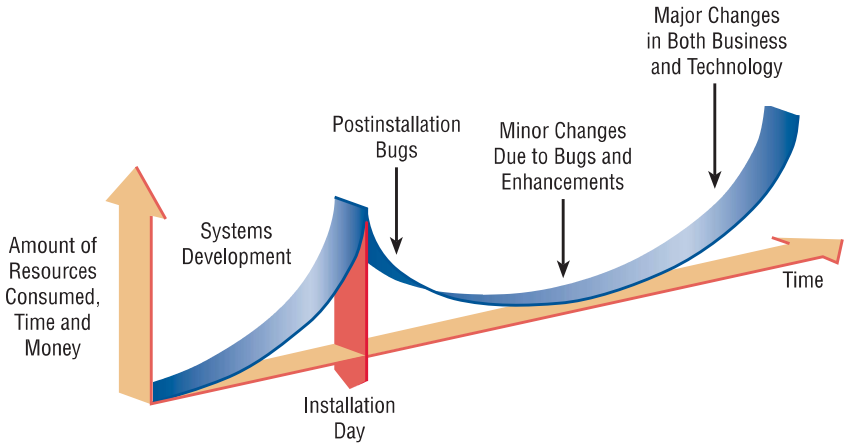


FIGURE 1.5

Resource consumption over the system life.

USING CASE TOOLS

Analysts who adopt the SDLC approach often benefit from productivity tools, called Computer-Aided Software Engineering (CASE) tools, that have been created explicitly to improve their routine work through the use of automated support. Analysts rely on CASE tools to increase productivity, communicate more effectively with users, and integrate the work that they do on the system from the beginning to the end of the life cycle.

Visible Analyst (VA) is one example of a CASE tool that enables systems analysts to do graphical planning, analysis, and design in order to build complex client/server applications and databases. Visible Analyst and another software product called Microsoft Visio allow users to draw and modify diagrams easily.

Analysts and users alike report that CASE tools afford them a means of communication about the system during its conceptualization. Through the use of automated support featuring onscreen output, clients can readily see how data flows and other system concepts are depicted, and they can then request corrections or changes that would have taken too much time with older tools.

Some analysts distinguish between upper and lower CASE tools. An upper CASE tool allows the analyst to create and modify the system design. All the information about the project is stored in an encyclopedia called the CASE repository, a large collection of records, elements, diagrams, screens, reports, and other information (see Figure 1.6). Analysis reports may be produced using the repository information to show where the design is incomplete or contains errors. Upper CASE tools can also help support the modeling of an organization's functional requirements, assist analysts and users in drawing the boundaries for a given project, and help them visualize how the project meshes with other parts of the organization.

Lower CASE tools are used to generate computer source code, eliminating the need for programming the system. Code generation has several advantages: (1) the system can be produced more quickly than by writing computer programs; (2) the amount of time spent on maintenance decreases with code generation; (3) code can be generated in more than one computer language, so it is easier to migrate systems from one platform to another; (4) code generation provides a cost-effective way of tailoring systems purchased from third-party vendors to the needs of the organization; and (5) generated code is free of computer program errors.

THE AGILE APPROACH

Although this text tends to focus on SDLC, the most widely used approach in practice, at times the analyst will recognize that the organization could benefit from an alternative approach. Perhaps a systems project using a structured approach has recently failed, or perhaps the organizational subcultures, composed of several different user groups, seem more in step with an alternative method. We cannot do justice to these methods in a small space; each deserves and has inspired its own books and research. By mentioning these approaches here, however, we hope to help you become aware that under certain circumstances, your organization may want to consider an alternative or supplement to structured analysis and design and to the SDLC.

The agile approach is a software development approach based on values, principles, and core practices. The four values are communication, simplicity, feedback, and courage. We recommend that systems analysts adopt these values in all projects they undertake, not just when adopting the agile approach.

In order to finish a project, adjustments often need to be made in project management. In Chapter 6 we will see that agile methods can ensure successful completion of a project by adjusting the important resources of time, cost, quality, and scope. When these four control variables are properly included in the planning, there is a state of balance between the resources and the activities needed to complete the project.

Taking development practices to the extreme is most noticeable when one pursues practices that are unique to agile development. In Chapter 6 we discuss four core agile practices: short releases, the 40-hour workweek, hosting an onsite customer, and using pair programming. At first glance these practices appear extreme, but as you will see, we can learn some important lessons from incorporating many of the values and practices of the agile approach into systems analysis and design projects.

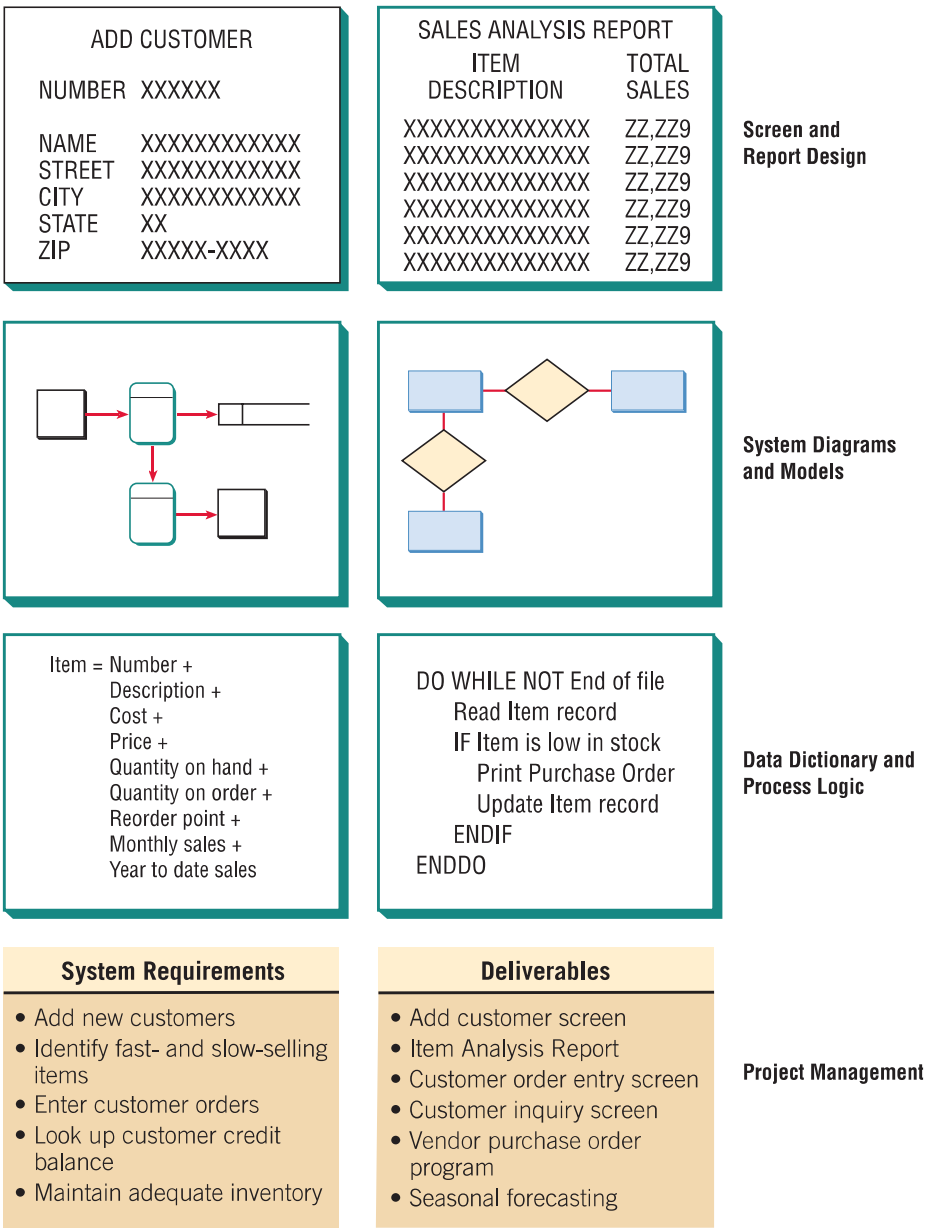


FIGURE 1.6

The repository concept.

Developmental Process for an Agile Project

There are activities and behaviors that shape the way development team members and customers act during the development of an agile project. Two words that characterize a project done with an agile approach are interactive and incremental. By examining Figure 1.7 we can see that there are five distinct stages: exploration, planning, iterations to the first release, productionizing, and maintenance. Notice that the three red arrows that loop back into the "Iterations" box symbolize incremental changes created through repeated testing and feedback that eventually lead to a stable but evolving system. Also note that there are multiple looping arrows that feed back into the productionizing phase. These symbolize that the pace of iterations is increased after a product is released. The red arrow is shown leaving the maintenance stage and returning to the planning stage, so that there is a continuous feedback loop involving customers and the development team as they agree to alter the evolving system.

EXPLORATION. During exploration, you will explore your environment, asserting your conviction that the problem can and should be approached with agile development, assemble the team, and assess team member skills. This stage will take anywhere from a few weeks (if you already know

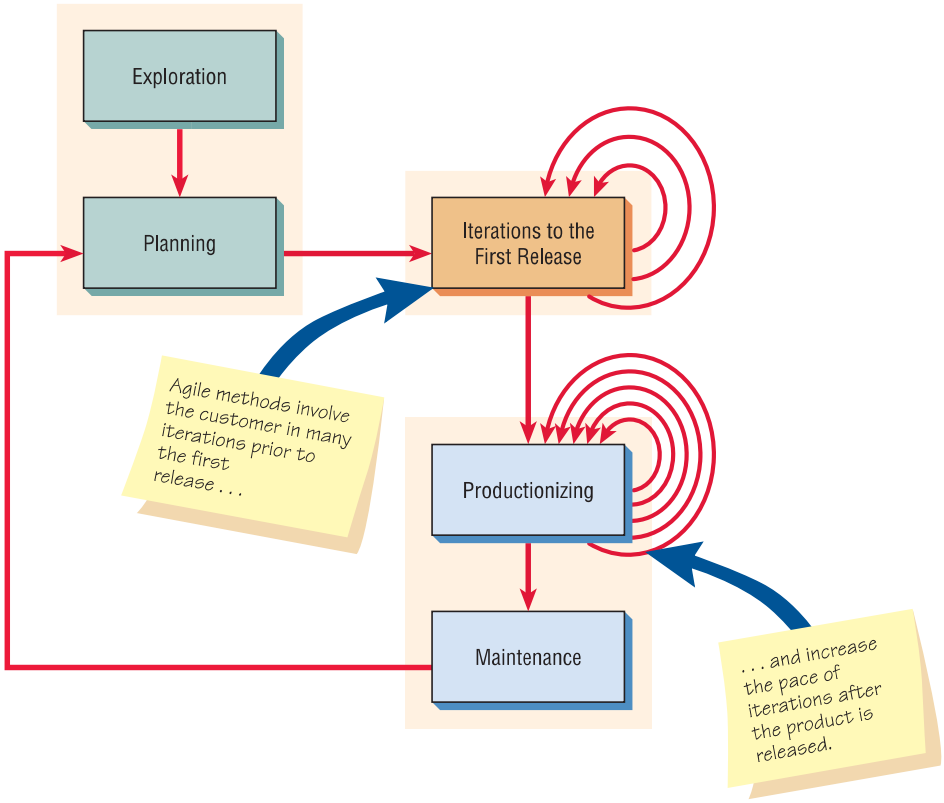


FIGURE 1.7

The five stages of the agile modeling development process show that frequent iterations are essential to successful system development.

your team members and technology) to a few months (if everything is new). You also will be actively examining potential technologies needed to build the new system. During this stage you should practice estimating the time needed for a variety of tasks. In exploration, customers also are experimenting with writing user stories. The point is to get the customer to refine a story enough so that you can competently estimate the amount of time it will take to build the solution into the system you are planning. This stage is all about adopting a playful and curious attitude toward the work environment, its problems, technologies, and people.

PLANNING. The next stage of the agile development process is called planning. In contrast to the first stage, planning may only take a few days to accomplish. In this stage you and your customers agree on a date anywhere from two months to half a year from the current date to deliver solutions to their most pressing business problems (you will be addressing the smallest, most valuable set of stories). If your exploration activities were sufficient, this stage should be very short.

The entire agile planning process has been characterized using the idea of a *planning game* as devised by Beck. The planning game spells out rules that can help formulate the agile development team's relationship with their business customers. Although the rules form an idea of how you want each party to act during development, they are not meant as a replacement for a relationship. They are a basis for building and maintaining a relationship.

So, we use the metaphor of a game. To that end we talk in terms of the goal of the game, the strategy to pursue, the pieces to move, and the players involved. The goal of the game is to maximize the value of the system produced by the agile team. In order to figure the value, you have to deduct costs of development, and the time, expense, and uncertainty taken on so that the development project could go forward.

The strategy pursued by the agile development team is always one of limiting uncertainty (downplaying risk). To do that they design the simplest solution possible, put the system into production as soon as possible, get feedback from the business customer about what's working, and adapt their design from there.

Story cards become the pieces in the planning game that briefly describe the task, provide notes, and provide an area for task tracking.

There are two main players in the planning game: the development team and the business customer. Deciding which business group in particular will be the business customer is not always

easy, because the agile process is an unusually demanding role for the customer to play. Customers decide what the development team should tackle first. Their decisions will set priorities and check functionalities throughout the process.

ITERATIONS TO THE FIRST RELEASE. The third stage in the agile development process is composed of iterations to the first release. Typically these are iterations (cycles of testing, feedback, and change) of about three weeks in duration. You will be pushing yourself to sketch out the entire architecture of the system, even though it is just in outline or skeletal form. One goal is to run customer-written functional tests at the end of each iteration. During the iterations stage you should also question whether the schedule needs to be altered or whether you are tackling too many stories. Make small rituals out of each successful iteration, involving customers as well as developers. Always celebrate your progress, even if it is small, because this is part of the culture of motivating everyone to work extremely hard on the project.

PRODUCTIONIZING. Several activities occur during this phase. In this phase the feedback cycle speeds up so that rather than receiving feedback for an iteration every three weeks, software revisions are being turned around in one week. You may institute daily briefings so everyone knows what everyone else is doing. The product is released in this phase, but may be improved by adding other features. Getting a system into production is an exciting event. Make time to celebrate with your teammates and mark the occasion. One of the watchwords of the agile approach, with which we heartily agree, is that it is supposed to be fun to develop systems!

MAINTENANCE. Once the system has been released, it needs to be kept running smoothly. New features may be added, riskier customer suggestions may be considered, and team members may be rotated on or off the team. The attitude you take at this point in the developmental process is more conservative than at any other time. You are now in a "keeper of the flame" mode rather than the playful one you experienced during exploration.

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN

Object-oriented (O-O) analysis and design is an approach that is intended to facilitate the development of systems that must change rapidly in response to dynamic business environments. Chapter 10 helps you understand what object-oriented systems analysis and design is, how it differs from the structured approach of the SDLC, and when it may be appropriate to use an object-oriented approach.

Object-oriented techniques are thought to work well in situations in which complicated information systems are undergoing continuous maintenance, adaptation, and redesign. Object-oriented approaches use the industry standard for modeling object-oriented systems, called the unified modeling language (UML), to break down a system into a use case model.

Object-oriented programming differs from traditional procedural programming by examining objects that are part of a system. Each object is a computer representation of some actual thing or event. Objects may be customers, items, orders, and so on. Objects are represented by and grouped into classes that are optimal for reuse and maintainability. A class defines the set of shared attributes and behaviors found in each object in the class.

The phases in UML are similar to those in the SDLC. Since those two methods share rigid and exacting modeling, they happen in a slower, more deliberate pace than the phases of agile modeling. The analyst goes through problem and identification phases, an analysis phase, and a design phase as shown in Figure 1.8. Although much of the specifics are discussed in Chapters 2 and 10, the following steps give a brief description of the UML process.

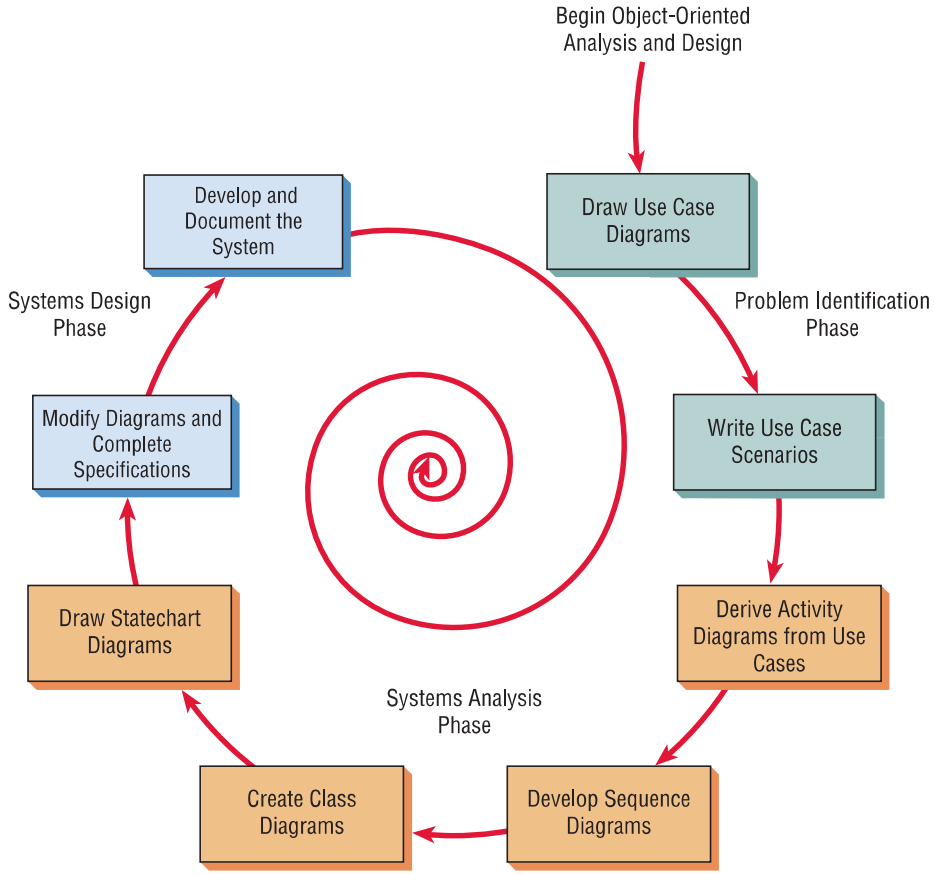
1. Define the use case model.

   In this phase the analyst identifies the actors and the major events initiated by the actors. Often the analyst will start by drawing a diagram with stick figures representing the actors and arrows showing how the actors relate. This is called a use case diagram (Chapter 2) and it represents the standard flow of events in the system. Then an analyst typically writes up a use case scenario (Chapter 2), which describes in words the steps that are normally performed.
2. During the systems analysis phase, begin drawing UML diagrams.

   In the second phase (Chapter 10), the analyst will draw Activity Diagrams, which illustrate all the major activities in the use case. In addition, the analyst will create one or

FIGURE 1.8

The steps in the UML development process.



more sequence diagrams for each use case, which show the sequence of activities and their timing. This is an opportunity to go back and review the use cases, rethink them, and modify them if necessary.

3. Continuing in the analysis phase, develop class diagrams.

   The nouns in the use cases are objects that can potentially be grouped into classes. For example, every automobile is an object that shares characteristics with other automobiles. Together they make up a class.

4. Still in the analysis phase, draw statechart diagrams.

   The class diagrams are used to draw statechart diagrams, which help in understanding complex processes that cannot be fully derived by the sequence diagrams. The statechart diagrams are extremely useful in modifying class diagrams, so the iterative process of UML modeling continues.

5. Begin systems design by modifying the UML diagrams. Then complete the specifications.

   Systems design means modifying the existing system and that implies modifying the diagrams drawn in the previous phase. These diagrams can be used to derive classes, their attributes, and methods (methods are simply operations). The analyst will need to write class specifications for each class including the attributes, methods, and their descriptions. They will also develop methods specifications that detail the input and output requirements for the method, along with a detailed description of the internal processing of the method.

6. Develop and document the system.

   UML is, of course, a modeling language. An analyst may create wonderful models, but if the system isn't developed there is not much point in building models. Documentation is critical. The more complete the information you provide the development team through documentation and UML diagrams, the faster the development and the more solid the final production system.

Object-oriented methodologies often focus on small, quick iterations of development, sometimes called the spiral model. Analysis is performed on a small part of the system, usually starting

Choose	When
The Systems Development Life Cycle (SDLC) Approach	• systems have been developed and documented using SDLC • it is important to document each step of the way • upper-level management feels more comfortable or safe using SDLC • there are adequate resources and time to complete the full SDLC • communication of how new systems work is important
Agile Methodologies	• there is a project champion of agile methods in the organization • applications need to be developed quickly in response to a dynamic environment • a rescue takes place (the system failed and there is no time to figure out what went wrong) • the customer is satisfied with incremental improvements • executives and analysts agree with the principles of agile methodologies
Object-Oriented Methodologies	• the problems modeled lend themselves to classes • an organization supports the UML learning • systems can be added gradually, one subsystem at a time • reuse of previously written software is a possibility • it is acceptable to tackle the difficult problems first

FIGURE 1.9

How to decide which development method to use.

with a high-priority item or perhaps one that has the greatest risk. This is followed by design and implementation. The cycle is repeated with analysis of the next part, design, and some implementation, and it is repeated until the project is completed. Reworking diagrams and the components themselves is normal. UML is a powerful modeling tool that can greatly improve the quality of your systems analysis and design and the final product.

CHOOSING WHICH SYSTEMS DEVELOPMENT METHOD TO USE

The differences among the three approaches described earlier are not as big as they seem at the outset. In all three approaches, the analyst needs to understand the organization first (Chapter 2). Then the analyst or project team needs to budget their time and resources and develop a project proposal (Chapter 3). Next they need to interview organizational members and gather detailed data by using questionnaires (Chapter 4) and sample data from existing reports and observe how business is currently transacted (Chapter 5). The three approaches have all of these activities in common.

Even the methods themselves have similarities. The SDLC and object-oriented approaches both require extensive planning and diagramming. The agile approach and the object-oriented approach both allow subsystems to be built one at a time until the entire system is complete. The agile and SDLC approaches are both concerned about the way data logically moves through the system.

So given a choice to develop a system using an SDLC approach, an agile approach, or an object-oriented approach, which would you choose? Figure 1.9 provides a set of guidelines to help you choose which method to use when developing your next system.

SUMMARY

Information can be viewed as an organizational resource just as humans are. As such, it must be managed carefully, just as other resources are. The availability of affordable computer power to organizations has meant an explosion of information, and consequently, more attention must be paid to coping with the information generated.

Systems analysts recommend, design, and maintain many types of systems for users, including transaction processing systems (TPS), office automation systems (OAS), knowledge work systems (KWS), and management information systems (MIS). They also create decision-oriented systems for specific users. These

HYPERCASE® EXPERIENCE 1

"Welcome to Maple Ridge Engineering, what we call MRE. We hope you'll enjoy serving as a systems consultant for us. Although I've worked here five years in different capacities, I've just been reassigned to serve as an administrative aide to Snowden Evans, the head of the new Training and Management Systems Department. We're certainly a diverse group. As you make your way through the company, be sure to use all your skills, both technical and people oriented, to understand who we are and to identify the problems and conflicts that you think should be solved regarding our information systems."

"To bring you up to date, let me say that Maple Ridge Engineering is a medium-sized medical engineering company. Last year, our revenues exceeded $287 million. We employ about 335 people. There are about 150 administrative employees as well as management and clerical staff like myself; approximately 75 professional employees, including engineers, physicians, and systems analysts; and about 110 trade employees, such as drafters and technicians."

"There are four offices. You will visit us through HyperCase in our home office in Maple Ridge, Tennessee. We have three other branches in the southern United States as well: Atlanta, Georgia; Charlotte, North Carolina; and New Orleans, Louisiana. We'd love to have you visit when you're in the area."

"For now, you should explore HyperCase using either Firefox, Safari, or Microsoft Internet Explorer."

"To learn more about Maple Ridge Engineering as a company or to find out how to interview our employees, who will use the systems you design, and how to observe their offices in our company, you may want to start by going to the Web site found at www.pearsonhighered.com/kendall. Then click on the link labeled **HyperCase.** At the HyperCase display screen, click on **Start** and you will be in the reception room for Maple Ridge Engineering. From this point, you can start consulting right away."

This Web site contains useful information about the project as well as files that can be downloaded to your computer. There is a set of Visible Analyst data files, and another set of Visio data files that match HyperCase. They contain a partially constructed series of data flow diagrams, entity-relationship diagrams, UML diagrams, and repository information. The HyperCase Web site also contains additional exercises that may be assigned. HyperCase is designed to be explored, and you should not overlook any object or clue on a Web page.

include decision support systems (DSS), expert systems (ES), group decision support systems (GDSS), computer-supported collaborative work systems (CSCWS), and executive support systems (ESS). Many applications are either originating on, or moving to, the Web to support ecommerce and many other business functions.

Systems analysis and design is a systematic approach to identifying problems, opportunities, and objectives; to analyzing human and computer-generated information flows in organizations; and to designing computerized information systems to solve a problem. Systems analysts are required to take on many roles in the course of their work. Some of these roles are (1) an outside consultant to business, (2) a supporting expert within a business, and (3) an agent of change in both internal and external situations.

Analysts possess a wide range of skills. First and foremost, the analyst is a problem solver, someone who enjoys the challenge of analyzing a problem and devising a workable solution. Systems analysts require communication skills that allow them to relate meaningfully to many different kinds of people on a daily basis, as well as computer skills. Understanding and relating well to users is critical to their success.

Analysts proceed systematically. The framework for their systematic approach is provided in what is called the systems development life cycle (SDLC). This life cycle can be divided into seven sequential phases, although in reality the phases are interrelated and are often accomplished simultaneously. The seven phases are identifying problems, opportunities, and objectives; determining human information requirements; analyzing system needs; designing the recommended system; developing and documenting software; testing and maintaining the system; and implementing and evaluating the system.

The agile approach is a software development approach based on values, principles, and core practices. Systems that are designed using agile methods can be developed rapidly. Stages in the agile development process are exploration, planning, iterations to the first release, productionizing, and maintenance.

A third approach to systems development is called object-oriented analysis design. These techniques are based on object-oriented programming concepts that have become codified in UML, a standardized modeling language in which objects that are created include not only code about data but also instructions about the operations to be performed on the data. Key diagrams help analyze, design, and communicate UML-developed systems. These systems are usually developed as components and reworking the components many times is a normal activity in object-oriented analysis and design.

KEYWORDS AND PHRASES

agent of change
agile approach
agile methods
Ajax
artificial intelligence (AI)
bespoke software
Computer-Assisted Software Engineering (CASE)
CASE tools
computer-supported collaborative work systems (CSCWS)
decision support systems (DSS)
ecommerce applications
enterprise resource planning (ERP) systems
executive support systems (ESS)
expert systems
exploration phase
group decision support systems (GDSS)
human–computer interaction (HCI)
iterations to the first release phase
knowledge work systems (KWS)
maintenance phase
management information systems (MIS)
mcommerce (mobile commerce)
migrate systems
object-oriented (O-O) systems analysis and design
office automation systems (OAS)
open source software (OSS)
planning game
planning phase
productionizing phase
prototyping
rapid application development (RAD)
service-oriented architecture (SOA)
systems analysis and design
systems analyst
systems consultant
systems development life cycle (SDLC)
transaction processing systems (TPS)
unified modeling language (UML)

REVIEW QUESTIONS

1. Compare treating information as a resource to treating humans as a resource.
2. List the differences between OAS and KWS.
3. Define what is meant by MIS.
4. How does MIS differ from DSS?
5. Define the term *expert systems.* How do expert systems differ from decision support systems?
6. List the problems of group interaction that group decision support systems (GDSS) and computer-supported collaborative work systems (CSCWS) were designed to address.
7. Which is the more general term, CSCWS or GDSS? Explain.
8. Define the term *mcommerce.*
9. List the advantages of mounting applications on the Web.
10. What is the overarching reason for designing enterprise (or ERP) systems?
11. Provide an example of an open source software project.
12. List the advantages of using systems analysis and design techniques in approaching computerized information systems for business.
13. List three roles that the systems analyst is called upon to play. Provide a definition for each one.
14. What personal qualities are helpful to the systems analyst? List them.
15. List and briefly define the seven phases of the systems development life cycle (SDLC).
16. What are CASE tools used for?
17. What is the difference between upper and lower CASE tools?
18. Define what is meant by the agile approach.
19. What is the meaning of the phrase "the planning game"?
20. What are the stages in agile development?
21. Define the term *object-oriented analysis* and *design.*
22. What is UML?

SELECTED BIBLIOGRAPHY

Coad, P., and E. Yourdon. *Object-Oriented Analysis,* 2d ed. Englewood Cliffs, NJ: Prentice Hall, 1991.

Davis, G. B., and M. H. Olson. *Management Information Systems: Conceptual Foundation, Structure, and Development,* 2d ed. New York: McGraw-Hill, 1985.

Feller, J., P. Finnegan, D. Kelly, and M. MacNamara. "Developing Open Source Software: A Community-Based Analysis of Research." In IFIP International Federation for Information Processing, Vol. 208, *Social Inclusion: Societal and Organizational Implications for Information Systems.* Edited by E. Trauth, D. Howcroft, T. Butler, B. Fitzgerald, and J. DeGross, pp. 261–278. Boston: Springer, 2006.

Kendall, J. E., and K. E. Kendall. "Information Delivery Systems: An Exploration of Web Push and Pull Technologies." *Communications of AIS,* Vol. 1, Article 14, April 23, 1999.

Kendall, J. E., K. E. Kendall, and S. Kong. "Improving Quality Through the Use of Agile Methods in Systems Development: People and Values in the Quest for Quality." In *Measuring Information Systems Delivery Quality.* Edited by E. W. Duggan and H. Reichgelt, pp. 201–222. Hershey, PA: Idea Group Publishing, 2006.

Laudon, K. C., and J. P. Laudon. *Management Information Systems,* 11th ed. Upper Saddle River, NJ: Pearson Prentice Hall, 2010.

Verma, S. "Software Quality and the Open Source Process." In *Measuring Information Systems Delivery Quality.* Edited by E. W. Duggan and H. Reichgelt, pp. 284–303. Hershey, PA: Idea Group Publishing, 2006.

www.visible.com/Products/index.htm. Last accessed March 23, 2009.

Yourdon, E. *Modern Structured Analysis.* Englewood Cliffs, NJ: Prentice Hall, 1989.

Zhang, P., J. Carey, D. Te'eni, and M. Tremaine. "Integrating Human–Computer Interaction Development into the Systems Development Life Cycle: A Methodology." *Communications of the Association for Information Systems,* Vol. 15, 2005, pp. 512–543.