Systems Analysis and Design

TENTH EDITION

Kenneth E. Kendall • Julie E. Kendall

CONSULTING OPPORTUNITIES

OTHER MIS TITLES OF INTEREST

Introductory MIS

Experiencing MIS, 8/e
Kroenke & Boyle ©2020

Using MIS, 10/e
Kroenke & Boyle ©2018

Management Information Systems, 16/e
Laudon & Laudon ©2020

Essentials of MIS, 13/e
Laudon & Laudon ©2019

Processes, Systems, and Information: An Introduction to MIS, 3/e
McKinney & Kroenke ©2019

Information Systems Today, 8/e
Valacich & Schneider ©2018

Introduction to Information Systems, 3/e
Wallace ©2018

Database

Hands-on Database, 2/e
Conger ©2014

Modern Database Management, 13/e
Hoffer, Ramesh & Topi ©2020

Database Concepts, 8/e
Kroenke, Auer, Vandenberg & Yoder ©2018

Database Processing, 15/e
Kroenke, Auer, Vandenberg & Yoder ©2019

Systems Analysis and Design

Modern Systems Analysis and Design, 8/e
Hoffer, George & Valacich ©2017

Decision Support Systems

Business Intelligence, Analytics, and Data Science, 4/e
Sharda, Delen & Turban ©2018

Business Intelligence and Analytics: Systems for Decision Support, 10/e
Sharda, Delen & Turban ©2014

Data Communications & Networking

Applied Networking Labs, 2/e
Boyle ©2014

Digital Business Networks
Dooley ©2014

Business Data Networks and Security, 11/e
Panko & Panko ©2019

Electronic Commerce

E-commerce 2019: Business. Technology. Society, 15/e
Laudon & Traver ©2020

Enterprise Resource Planning

Enterprise Systems for Management, 2/e
Motiwalla & Thompson ©2012

Project Management

Project Management: Process, Technology and Practice
Vaidyanathan ©2013

SYSTEMS ANALYSIS AND DESIGN

This page is intentionally left blank.

TENTH EDITION
GLOBAL EDITION

SYSTEMS ANALYSIS AND DESIGN

Kenneth E. Kendall
RUTGERS UNIVERSITY
School of Business–Camden
Camden, New Jersey

Julie E. Kendall
RUTGERS UNIVERSITY
School of Business–Camden
Camden, New Jersey

Pearson

Harlow, England • London • New York • Boston • San Francisco • Toronto • Sydney • Dubai • Singapore • Hong Kong
Tokyo • Seoul • Taipei • New Delhi • Cape Town • Sao Paulo • Mexico City • Madrid • Amsterdam • Munich • Paris • Milan

Vice President, IT & Careers: Andrew Gilfillan
Senior Portfolio Manager: Samantha Lewis
Managing Producer: Laura Burgess
Associate Content Producer: Stephany Harrington
Portfolio Management Assistant: Madeline Houpt
Assistant Acquisitions Editor, Global Edition: Rosemary Iles
Associate Project Editor, Global Edition: Aurko Mitra
Content Producer, Global Edition: Sonam Arora
Director of Product Marketing: Brad Parkins
Product Marketing Manager: Heather Taylor
Product Marketing Assistant: Jesika Bethea
Field Marketing Manager: Molly Schmidt
Field Marketing Assistant: Kelli Fisher
Cover Image: LIUSHENGFILM/Shutterstock
Vice President, Product Model Management: Jason Fournier
Senior Product Model Manager: Eric Hakanson
Lead, Production and Digital Studio: Heather Darby
Digital Studio Course Producer: Jaimie Noy
Media Production Manager, Global Edition: Vikram Kumar
Senior Manufacturing Controller, Global Edition: Kay Holman
Program Monitor: Freddie Domini, SPi Global
Full-Service Project Management: Cenveo® Publisher Services
Cover Designer, Global Edition: Lumina Datamatics Inc.

Pearson Education Limited
KAO Two
KAO Park
Hockham Way
Harlow
Essex
CM17 9SR
United Kingdom

and Associated Companies throughout the world

Visit us on the World Wide Web at: www.pearsonglobaleditions.com

Authorized adaptation from the United States edition, entitled *Systems Analysis and Design*, 10th Edition, ISBN 978-0-13-478555-4, by Kenneth E. Kendall and Julie E. Kendall, published by Pearson Education © 2019.

ISBN 10: 1-292-28145-6
ISBN 13: 978-1-292-28145-2

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library

Typeset by Cenveo® Publisher Services

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

This page is intentionally left blank.

BRIEF CONTENTS

This page is intentionally left blank.

CONTENTS

PREFACE

NEW TO THIS EDITION

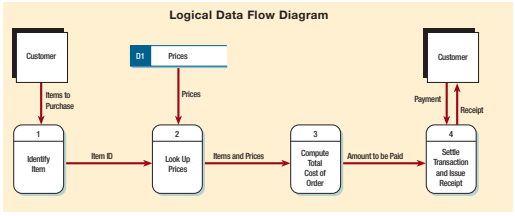
The tenth edition of *Systems Analysis and Design* includes extensive changes inspired by the rapid changes in the IS field over the past four years, and they are included as a response to the thoughtful input of our audience of adopters, students, and academic reviewers. Many new and advanced features are integrated throughout this new edition. In particular:

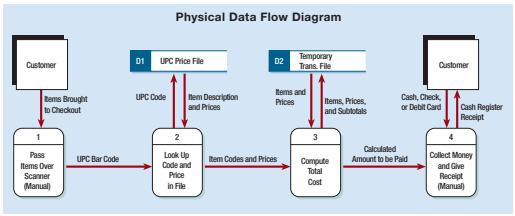
- Innovative material on using responsive design to enable systems analysts and organizations to participate in open source communities (Chapter 1)
- New coverage of the importance of security considerations right from the outset of a systems project (Chapter 1)
- New material on work-sanctioned social media sites to create productive systems development subcultures and collaborative design (Chapter 2)
- Additional material on cloud computing as a platform choice for a systems development project (Chapter 3)
- Innovative material on listening to user stories to complement other interactive information gathering methods (Chapter 4)
- New material on text analytics software to examine unstructured, soft data from customers' blogs, wikis, and social media sites to interpret qualitative material (Chapter 5)
- New and expanded content on agile methods including Scrum, Scrum planning poker, the product backlog, sprint cycle, and burndown charts (Chapter 6)
- New coverage on Kanban systems as they apply to software development (Chapter 6)
- Innovative coverage of DevOps as a cultural shift in the way to organize rapid systems development and operations (Chapter 6)
- Additional material on designing dashboards for decision makers using infographics (Chapter 11)
- New material on responsive Web design for websites that can be viewed on any device (Chapter 11)
- New material comparing and contrasting skeuomorphic design with flat design for websites (Chapter 11)
- Additional material on innovative guidelines for designing for social media (Chapter 11)
- New content on website design including use of navigational elements such as a hamburger icon and breadcrumb trail (Chapter 12)
- New material on the relationship of business intelligence to data warehouses, big data, and data analytics (Chapter 13)
- Additional coverage on database security and risk tradeoffs in securing databases (Chapter 13)
- Innovative material on developing and using blockchains to provide a verifiable electronic record for tracking any kind of business asset (Chapter 13)
- New content on UX design (user experience design) for developing customer-centered ecommerce website experiences (Chapter 14)
- Innovative coverage of designing virtual reality, augmented reality, and intelligent personal assistants (Chapter 14)
- Additional content on using QR codes for improved data entry (Chapter 15)
- Additional material on designing improved cloud security, privacy, and stability, especially for business continuity and disaster recovery (Chapter 16)

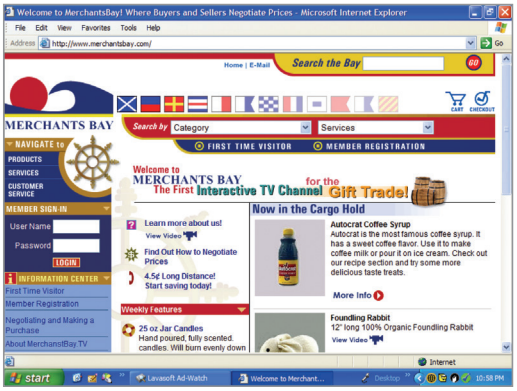
DESIGN FEATURES

Figures have a stylized look 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 students can easily distinguish 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. In this edition we introduce UX (user experience) design. Screens are of the utmost importance when we put the user experience first. 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 how to capture user stories, as well as for input and output design and 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 hand. Although most organizations have computerization of manual processes as their goal, much data capture is still done using hand-written paper forms. Improved form design enables analysts to ensure accurate and complete input and output. Better forms also streamline new internal workflows that result from newly automated business-to-consumer (B2C) applications for ecommerce on the Web.

Need or Opportunity:	Apply shortcut methods for faster checkout.					
Story:	If the identity of the customer is known and the delivery address matches, speed up the transaction by accepting the credit card on file and the rest of the customer's preferences such as shipping method.					
		Well Below	**Below Average**	**Average**	**Above Average**	**Well Above**
Activities:	**Coding**			✓		
	Testing			✓		
	Listening			✓		
	Designing				✓	
Resources:	**Time**				✓	
	Cost		✓			
	Quality				✓	
	Scope			✓		

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

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

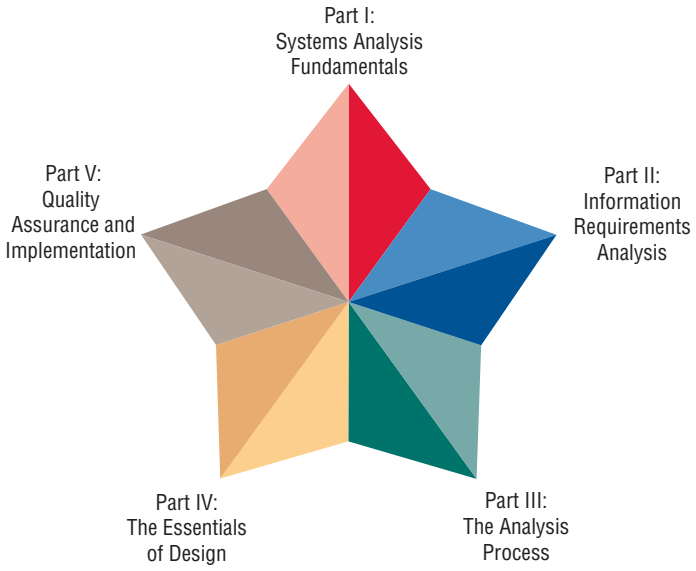
Break these down further,

then estimate time required.

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. This book is built on the idea 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, whether they are posed by designing for multiple platforms, new types of users, or implementing cloud-based systems; and by keeping up to date in their profession through the application of new methods, software, and alternative tools.

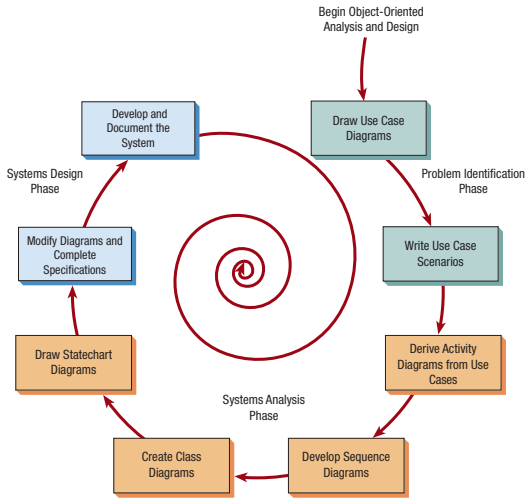
A BRIEF TOUR OF THE TENTH EDITION



Systems analysis and design is typically taught in one or two semesters. This 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, the legacy CPU Case online, or other materials available at the Pearson Instructor Resources website.

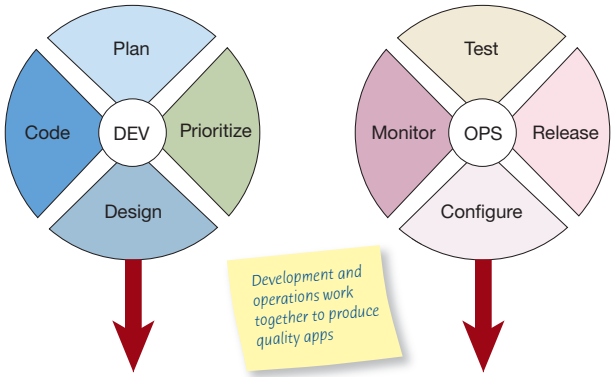
The text is divided into five major parts: Systems Analysis Fundamentals (Part I), Information Requirements Analysis (Part II), The Analysis Process (Part III), The Essentials of Design (Part IV), and Quality Assurance and Implementation (Part V).

Part I (Chapters 1–3) stresses the basics 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 universal modeling language (UML), along with reasons and situations for when to use them. Part I introduces the three roles of a systems analyst—consultant, supporting expert, and agent of change—along with ethical issues and professional guidelines for serving as a systems consultant. The importance of designing security into new systems from the beginning is noted. Material on virtual teams and virtual organizations, and the concept of human–computer interaction (HCI) is introduced as well. The use of open source software (OSS) and how analysts and organizations can participate in open source communities by using responsive design is introduced.

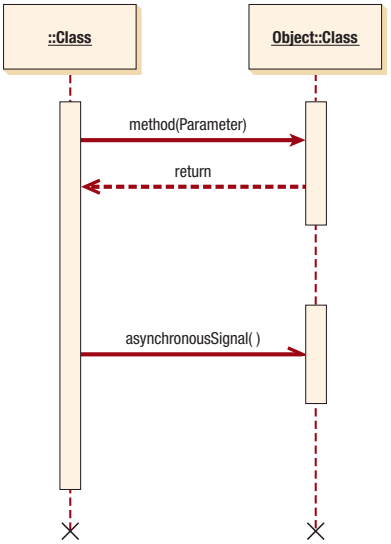


Chapter 2 includes how to initially approach an organization by drawing context-level data flow diagrams, using entity-relationship models, and developing use cases and use case scenarios. It views the organization as a system through the description of enterprise resource planning (ERP) systems. Also included is the importance of using employer-sanctioned social media to create a strong workplace culture. Chapter 3 focuses on project management. It introduces material on when to use cloud services versus purchasing hardware and software. Project management techniques including time estimation techniques for project management are discussed. Material in Chapter 3 will help students approach projects using the work breakdown structure (WBS). Creating a problem definition, developing a project charter, and determining feasibility are also covered. Chapter 3 guides students in writing and presenting a professional and effective systems proposal 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 a system. Chapter 4 introduces a group of interactive methods, including interviewing, joint application design (JAD), and constructing questionnaires. It expands material on listening to user stories in order to understand organization behaviors and values. 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. New material on the use of text analytics software to examine unstructured data from blogs, wikis, interviews, and social media sites is added. 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, and 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. New and expanded coverage of agile methods including Scrum, Scum planning poker, the product backlog, sprint cycle, and burndown charts are included. Kanban systems as they apply to software development are introduced, and innovative coverage of DevOps as a cultural shift in the way to organize rapid systems development and operations is covered.



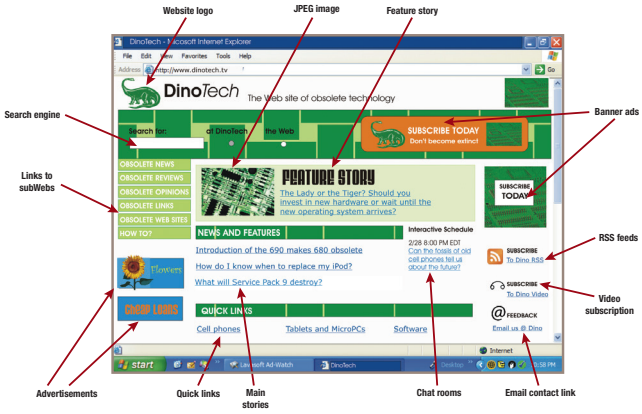
Part III (Chapters 7–10) details the analysis process. It builds on the previous two parts to move students into analysis of data flows as well as structured and semistructured decisions. It provides step-by-step details on how to use structured techniques to draw data flow diagrams (DFDs). Chapter 7 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.

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 considers output, including Web displays, audio, and electronic output such as web pages, email, and RSS feeds. Designing a website for ecommerce purposes is emphasized, and the importance of adding Web 2.0 technologies and social media to corporate and ecommerce websites is explored. Additional material on designing dashboards using infographics is provided. New material on responsive Web design is included. Flat versus skeuomorphic design is covered, as are innovative guidelines for designing for social media. Designing apps for smartphones and tablets is integrated, along with storyboarding, wireframing, and mockups. Output production and XML are covered.



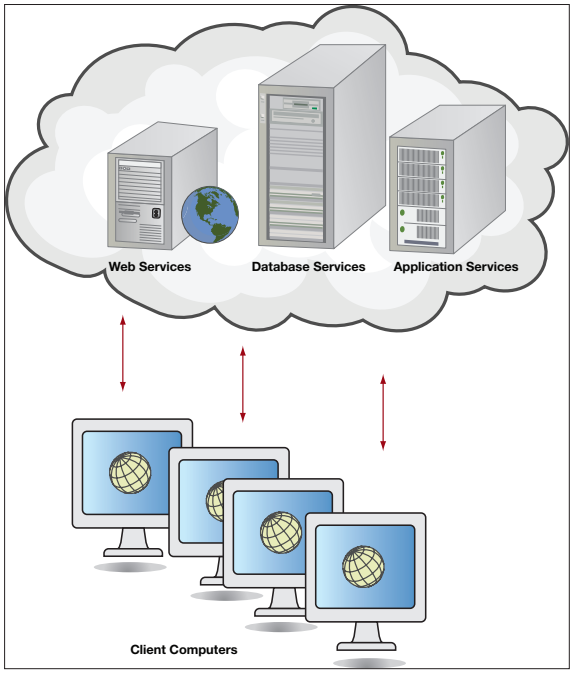
Chapter 12 includes innovative material on designing for smartphones and tablets as well as designing Web-based input forms and other electronic forms design. Also included is computer-assisted forms design. Chapter 12 also features in-depth coverage of website design, new content on how to add navigational elements to websites such as a hamburger menus and breadcrumb trails, and includes guidelines on when designers should add video, audio, and animation to website designs. There is detailed consideration of how to create effective graphics for corporate websites and ways to design effective onscreen navigation for website users.

Coverage of intranet and extranet page design is included. Consideration of database integrity constraints and how the user interacts with the computer and how to design an appropriate interface are discussed. 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 an entity-relationship diagram to determine record keys and provides 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. New material on the relationship between business intelligence (BI) and data warehouses, big data, and data analytics software is added in the context of data warehouses. Additional material on database security and risk tradeoffs in securing databases is added. Innovative material on developing and using blockchains to provide a verifiable electronic record for tracking any kind of business asset is included.

Chapter 14 emphasizes human–computer interaction (HCI), especially as it relates to interface design, as well as UX design. It discusses the importance of HCI in designing systems that suit individuals and assisting them in achieving personal and organization goals through their use of information technology. The concept of usability is introduced, so that systems analysis students can knowledgeably incorporate HCI practices in their designs. Chapter 14 introduces material on how to design gesture-based (multitouch) interfaces for smartphones and tablets, as well as designing alerts, notices, and queries. Material on designing easy onscreen navigation for website visitors is included. The chapter presents innovative approaches to searching on the Web, highlights material on graphic user interface (GUI) design, and provides innovative approaches to designing dialogues. Chapter 14 articulates specialized design considerations for ecommerce websites. New material on UX design (user experience design) for developing customer-centered ecommerce websites is included. Mashups, new applications created by combining two or more Web-based application programming interfaces, are also covered. Innovative material on designing virtual reality (VR), augmented reality (AR), and intelligent personal assistants is included. Chapter 14 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. It includes suggestions for incorporating two-dimensional codes QR codes and bar codes into data entry designs. It also considers the usefulness of RFID for automatic data collection. 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 combined with ERP are significantly altering the landscape 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 includes security for firewalls, gateways, public key infrastructure (PKI), secure electronic transaction (SET), secure sockets layer (SSL), virus protection software, URL filtering products, email filtering products, and virtual private networks (VPN) is included. Additional coverage on designing improved cloud security, privacy, and stability, especially for business continuity and disaster recovery, is included.

Important coverage of how the analyst can promote and monitor a corporate website is included in this section, which features Web activity monitoring, website 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.

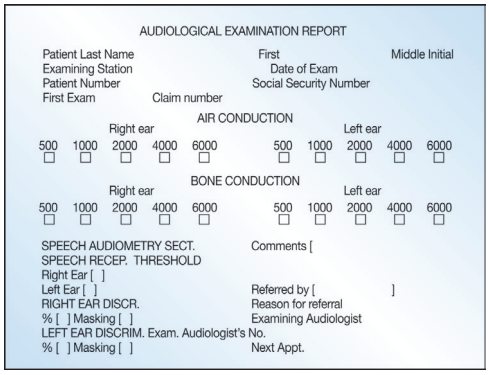
This tenth edition contains an updated **Glossary** of terms and a separate list of updated **Acronyms** used in the book and in the systems analysis and design field.

PEDAGOGICAL FEATURES

Chapters in this tenth edition contain:

- **Learning Objectives** at the beginning of each chapter
- **Summaries** at the end of each chapter that tie together the salient points of the chapter and provide an excellent source of review for exams
- **Keywords and Phrases** for each chapter
- **Review Questions** to help with learning key definitions and terms
- **Problems** that help students apply and extend the concepts and tools they are learning to practical situations
- **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 with more than 50 minicases throughout the book
- **Mac Appeal** columns that inform students about design software available on the Mac and iPhone
- **HyperCase Experiences** in each chapter simulate organizational experience and focus learning from HyperCase online

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: 500 ☐ 1000 ☐ 2000 ☐ 4000 ☐ 6000 ☐
Left ear: 500 ☐ 1000 ☐ 2000 ☐ 4000 ☐ 6000 ☐

BONE CONDUCTION

Right ear: 500 ☐ 1000 ☐ 2000 ☐ 4000 ☐ 6000 ☐
Left ear: 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.

This tenth edition presents more than 50 Consulting Opportunities, addressing significant and emerging topics arising in information systems, including designing systems from an HCI perspective, ecommerce applications for the Web, cloud computing decisions, and using UML to model information systems from an object-oriented perspective. Consulting Opportunities can be used for motivating thoughtful in-class discussions or assigned as homework or take-home exam questions.

Not all systems work demands extended two- or three-year projects, so many Consulting Opportunities included can be solved in 20 to 30 minutes of group discussion, group writing, 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. Mr. Evans 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. Complete a computer equipment inventory for the Training and Management Systems Department, describing all the systems you find. *Hint:* Create an inventory form to simplify your task.
2. Using the software evaluation guidelines given in the chapter, do a brief evaluation of GEMS, a software package used by the Training and Management Systems Department employees. In a paragraph, briefly critique this custom software by comparing it with COTS 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 Mr. Evans is considering for the proposed project tracking and reporting system.
5. What organizational and political factors should Mr. Evans consider in proposing his new system at MRE? (In a brief paragraph, discuss three central conflicts.)

Reception Area - Mozilla Firefox

MRE

EXIT

FIGURE 3.HC1

The reception room resembles a typical corporation. While you are in this HyperCase screen, click on the small red directory if you want to visit an MRE employee.

HyperCase Experiences that pose challenging student exercises are present in each chapter. HyperCase 2.10 has organization problems featuring information systems technology. HyperCase represents an original virtual organization that allows students who access it to become immediately immersed in organization life. Students will interview people, observe office environments, analyze their prototypes, and review the documentation of their existing systems.

HyperCase 2.10 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 a 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.

Each chapter contains HyperCase Experiences that include assignments (and even some clues) to help students solve difficult organization problems including developing new systems, merging departments, hiring 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.

EXPANDED WEB SUPPORT

Systems Analysis and Design, tenth edition, features Web-based support for solid but lively pedagogical techniques in the information systems field:

- The website, located at **www.pearsonglobaleditions.com**, contains a wealth of critical learning and support tools, which keep class discussions exciting.
- **HyperCase 2.10** is an award-winning, interactive organization game. Students are encouraged to interview people in the organization, analyze problems, drill down into and modify data flow diagrams and data dictionaries, react to prototypes, and design new input and output.
- **A legacy case, the Central Pacific University (CPU) case is online**. In keeping with our belief that a variety of approaches is important, the entire legacy Central Pacific University (CPU) case, accompanied by partially solved Student Exercises, is fully available online. The legacy CPU case makes use of Microsoft Access, Microsoft Visio, and the popular CASE tool Visible Analyst by Visible Systems, Inc., for the sample screen shots and the student exercises. The legacy CPU case takes students through all phases of the systems development life cycle.

EXPANDED INSTRUCTOR SUPPLEMENTAL WEB SUPPORT

Extended support for instructors using this edition can be found at the official website located at **www.pearsonglobaleditions.com**. Resources include:

- **Instructor's Manual**—The Instructor's Manual contains answers to problems, solutions to cases, and suggestions for approaching the subject matter.
- **PowerPoint Presentations—**The PowerPoints feature lecture notes that highlight key text terms and concepts. Professors can customize the presentation by adding their own slides or by editing the existing ones.
- **TestGen Testbank File**—The TestGen Testbank file is an extensive set of multiple-choice, true/false, and essay-type questions for each chapter of the text. Questions are ranked according to difficulty level and referenced with page numbers from the text. The TestGen Testbank file is available in Microsoft Word format and as the computerized Prentice Hall TestGen software, with course management system conversions.

- **TestGen Testbank—**Pearson Education's test-generating software is available from the **TestGen website.** The software is PC/Mac and Blackboard compatible and preloaded with all the Test Gen Testbank questions. You can manually or randomly view test questions and drag and drop to create a test. You can add or modify test-bank questions as needed.
- **Image Library—**This collection of the text art is organized by chapter. This collection includes all the figures, tables, and screenshots from the book. These images can be used to enhance class lectures and PowerPoint slides.
- **Solutions to the legacy CPU Case and Student Files**—These exercises are based on the legacy CPU case, with solutions and examples stored in Visible Analyst files and Microsoft Access files.
- **eBook of *Systems Analysis and Design, 10th edition, Global Edition*** available at **mypearsonstore.com**

Arco Iris de Colores (Rainbow Colors) by *Pedro Fuller*

ACKNOWLEDGMENTS

The field of information systems was changing swiftly as we were writing the tenth edition of *Systems Analysis and Design*. We are excited that this edition is being published at the right moment for us to capture many of these innovations in systems analysis and design.

One notable change is the practice of UX design for developing customer-centered ecommerce website experiences. In this process, a systems analyst observes the behavior of customers and strives to enhance customer satisfaction and loyalty. The analyst accomplishes this by improving usability and ease of use. UX design is a design culture that chooses to give the user a good experience over maximizing short-term profit. Hand in hand with this approach is the accelerating use of responsive website design (RWD) enabling Web-based information systems to adapt and display correctly on whatever device is used to view content. In addition, cloud computing and software as a service (SaaS) are effectively altering the way analysts need to approach designing systems solutions.

Throughout the book you will learn and apply numerous techniques, methods, tools, and approaches to help visually capture a system. 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 and your values as a systems analyst and a human being.

The artist, Pedro Fuller, who created the inspiring oil painting *Arco Iris de Colores* (*Rainbow Colors*) featured on the previous page and throughout the 10th edition, states, "Every piece I complete expresses something unique. I always give careful consideration to the way color and form work together. Some of the themes in my art are spirituality, music, politics, and happiness. Because I value music for its relaxing qualities, there is a visual rhythm in my work through which I try to reach this same calmer, more spiritual place." Pedro was born in Managua, Nicaragua, and as a teenager moved with his family to Camden, New Jersey. We hope that you as a student will strive to create something unique working with color and form as you learn to design screens, forms, websites, and expressions on social media.

It is, in fact, our own students who deserve recognition for this new edition because of their comments and suggestions for enhancements and their desire for increased depth in timely topics. Students told us that they quickly put to use the new material on agile methods, especially Scrum, UX design, as well as the material on DevOps. We want to thank our coauthor, Allen Schmidt, who worked with us on HyperCase 2.10 for all his support and collaboration over the years. He is an outstanding person. Our appreciation also goes to Peter Schmidt and Jason Reed for their improvements to the early HyperCase. In addition, we want to thank the other two original authors of HyperCase, Richard Baskerville and Raymond Barnes, who contributed so much to our lives and our projects over the years and are exceptional friends.

We would like to thank our tenth edition production team, especially the Pearson Senior Portfolio Manager of IT & MIS, Samantha Lewis, whose good humor and optimistic approach encouraged us to keep working. We are also grateful to Neha Bhargava, extremely capable project manager for her composed competency and for her enthusiasm in keeping the project going. Freddie C. Domini, our Program Monitor also deserves thanks for helping us succeed in making this a strong, comprehensive, and systematic revision. Their help and intense interest in our book facilitated the completion of this project in an apt and timely manner.

We also appreciate the encouragement and support of the entire Rutgers community, including Chancellor Phoebe A. Haddon, Dean Jaishankar Ganesh, and our colleagues and staff in the

Julie and Ken Kendall personally thank all of our friends in the theatre and the performing arts. Here are the Kendalls at the 2012 Tony Awards afterparty with Tony-Award winning Actor James Corden (*right*). Photo by Anita & Steve Shevett.

School of Business–Camden and throughout all of Rutgers. They have been very enthusiastic about this edition as well as the many translations and versions of *Systems Analysis and Design* available in Spanish, Chinese, English for the Indian subcontinent, and Indonesian.

All the reviewers for the tenth edition deserve our thanks as well. Their thoughtful feedback and recommendations helped to strengthen the book. They are:

Daniel Asamoah, Wright State University
Eralda Caushaj, Lawrence Technological University
George Cognet, Delaware Technical Community College
Jim Connolly, Canisius College
Henry J. Felch, University of Maine at Augusta
Brian Jones, Tennessee Tech University
Sarah Khan, North Carolina State University
Brenda Mak, San Francisco State University
Randie Mondoro, Raritan Valley Community College
Fay Cobb Payton, North Carolina State University
Mary Reed, University of Jamestown
Paul A. Seibert, North Greenville University
Wayne Spies, Mercy College
Laura Trevino, The University of Texas at El Paso
Merrill Warkentin, Mississippi State University

Many of our colleagues and friends have encouraged us throughout the process of writing this book. We thank them for their comments on our work. They include: Ayman Abu Hamdieh, Macedonio Alanis, the Ciupeks, Gordon Davis, Tim DiVito, John Drozdal, EgoPo, Rich and Margarita Elias, Matt Germonprez, Nancy V. Gulick, Andy Hamingson, Blake Ives, Colleen Kelly-Lawler, Ken and Jane Laudon, Josh Lawler, Kin Lee, Matt Levy, Lars Mathiassen, Joel and

Bobbie Porter, Caryn Schmidt, Marc and Jill Schniederjans, Gabriel Stelian-Shanks, the Vargos, Merrill Warkentin, Brian Warner, Jeff and Bonnie Weil, Arlene and Paul Wolfling, Brett Young, and all of our friends and colleagues in The Drama League, The Actors Fund, the American Theatre Wing, Azuka Theatre, The KPMG Foundation, The New York Marriott Marquis, the Association for Information Systems, the Decision Sciences Institute, IFIP Working Group 8.2, and all those involved in the PhD Project, which serves minority doctoral students in information systems.

ACKNOWLEDGMENTS FOR THE GLOBAL EDITION

Pearson would like to thank and acknowledge the following people for their contributions to the Global Edition.

Contributor
Sahil Raj, Punjabi University

Reviewers
Petter Dessne, University of Borås
Floriana Grasso, University of Liverpool
Kamran Munir, University of the West of England
Elias Pimenidis, University of the West of England

This page is intentionally left blank.

SYSTEMS ANALYSIS AND DESIGN

This page is intentionally left blank.

1 Systems, Roles, and Development Methodologies

LEARNING OBJECTIVES

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

1. Understand the need for systems analysis and design in organizations.
2. Realize what the many roles of a systems analyst are.
3. Comprehend the fundamentals of three development methodologies: the systems development life cycle (SDLC), the agile approach including Scrum, and object-oriented systems analysis and design.

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 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. There is usually 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.

Need for Systems Analysis and Design

Systems analysis and design, as performed by systems analysts, is used to understand what people need to be able to systematically analyze data input or data flow, process or transform data, store data, and output information in the context of a particular organization or enterprise. Systems analysts seek to identify and solve the right problems by performing a thorough analysis of the client's system. Furthermore, systems analysis and design is used to analyze, design, and implement improvements to the computerized information systems that support users and the functions of businesses.

Security is critical to the functioning of organizational information systems, and security is a challenge for everyone involved in systems development. Information systems present multiple vulnerabilities, and the idea of perfect security is a fantasy. Rather, organizations make trade-offs, weighing the value of the data they are storing with the risk that they will experience a security breach. As a systems developer, it is important that you design new systems with an awareness of what you can do to design security into a system from the very beginning. Developing privacy controls and security by design, from the outset of systems design, is much more desirable and effective than adding it to older, legacy systems. Of course, you should always examine systems you are updating or improving for ways to address vulnerabilities and improve training for security issues.

Developing 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 a 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.

Roles of a Systems Analyst

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

Systems Analyst as Consultant

A 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 bring with them a fresh perspective that 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 organization 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.

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 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 website 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 be 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 coding 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. 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 coder." Then she ponders aloud, "I wonder how important unified modeling language (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?

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, an analyst draws on professional expertise concerning computer hardware and software and their uses in the business. This work is often not a full-blown systems project and may entail only a small modification or a 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 managing it. 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).

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.

Qualities of a Systems Analyst

From the foregoing descriptions of the roles the systems analyst plays, it is easy to see that a successful systems analyst must possess a wide range of qualities. Above all, an 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, an analyst must be able to systematically tackle the situation at hand through a skillful application of tools, techniques, and experience.

An 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 when interacting with technology, and they need enough computer experience to code, to understand the capabilities of computers, to glean information requirements from users, and to communicate what is needed to coders. Analysts also need to possess strong personal and professional ethics to help them shape their client relationships.

A 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, it is an ever changing and always challenging one.

The Systems Development Life Cycle

Throughout this chapter, we refer to the systematic approach analysts take to the analysis and design of information systems. Much of this is embodied in the three approaches we describe in this book. The first approach is called the systems development life cycle (SDLC). The SDLC is a phased approach to analysis and design based on the assumption that systems are best developed through the use of a specific cycle of analyst and user activities. It has also been called the waterfall method because the system analysis completes the first phase, then moves down to the next, and so on, like water flowing steadily downward from one rock to another.

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 1.1. Although each phase is presented discretely, it is never accomplished as a separate step. Instead, several activities may occur simultaneously, and activities may be repeated.

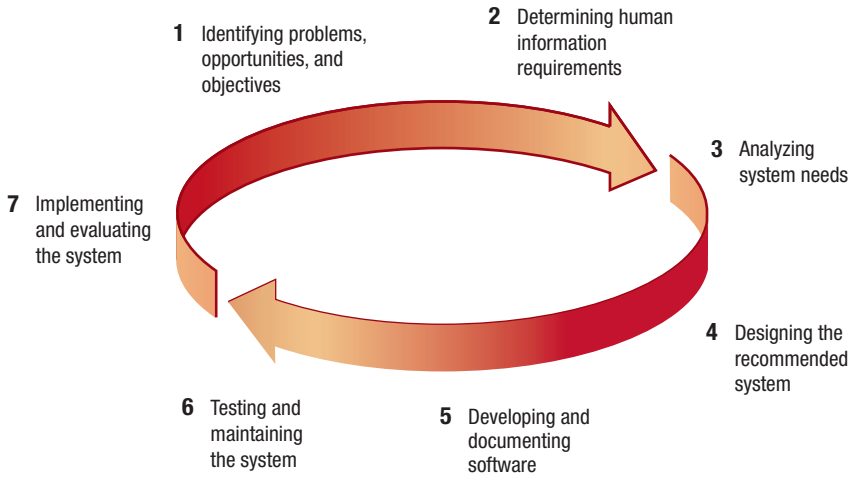


FIGURE 1.1

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

Identifying Problems, Opportunities, and Objectives

In this first phase of the SDLC, an 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 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 organization members, the analyst pinpoints problems. Others often bring up these problems, and they are the reason the analyst was initially called. 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 that contains a problem definition and summarizes 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 if it 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

In the next phase, the analyst determines 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 uses interactive methods such as interviewing, sampling and investigating hard data, and using questionnaires; 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 to be done? What new tasks are enabled by the new system that users were unable to do without it? How can the new system 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 outgrowth of the 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 OOAs 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 (DFDs) chart the input, processes, and output of the business's functions, and activity diagrams or sequence diagrams show the sequence of events, illustrating systems in a structured, graphic 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. Three major tools are used when analyzing structured decisions: structured English, decision tables, and decision trees.

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 of the Mac options available to a systems designer. Today, 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 Fusion by VMware, 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.

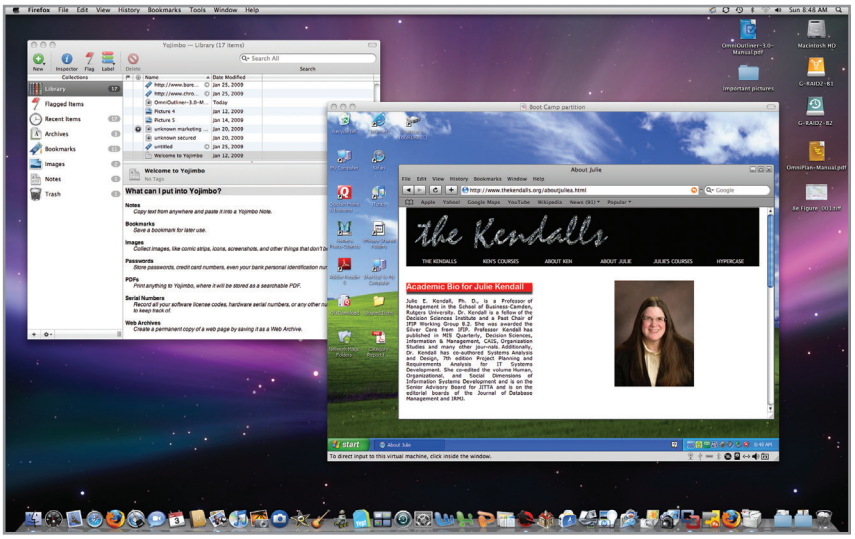


FIGURE 1.MAC

Running Windows on a Mac using virtualization called VM Fusion. (Screen shot reprinted with permission from Apple Inc. and Bare Bones Software)

At this point in the SDLC, the systems analyst prepares a systems proposal that summarizes what has been discovered 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 extremely important. The user interface is designed with the help of users to make sure 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 graphic 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.

Developing and Documenting Software

In the fifth phase of the SDLC, the analyst works with coders 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 websites featuring frequently asked questions (FAQs) or Read Me files shipped with new software. Because users are involved from the beginning, 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.

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

Testing and Maintaining the System

Before an information system can be used, it must be tested. It is much less costly to catch problems before rather than after the system is signed over to users. Coders alone complete some of the testing, and some testing is done by systems analysts in conjunction with coders. A series of tests to pinpoint problems is run, first with sample data and eventually with actual data from the current system. Test plans often 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 coder'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.

Evaluation is included as part of this final phase of the SDLC, but in fact 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 applications must be modified and kept up to date. Estimates of the time spent by departments on maintenance have ranged from 48 percent 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 a system is tested, bugs or errors creep into computer applications. Bugs in commercial PC software are often documented as "known anomalies," and they 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:

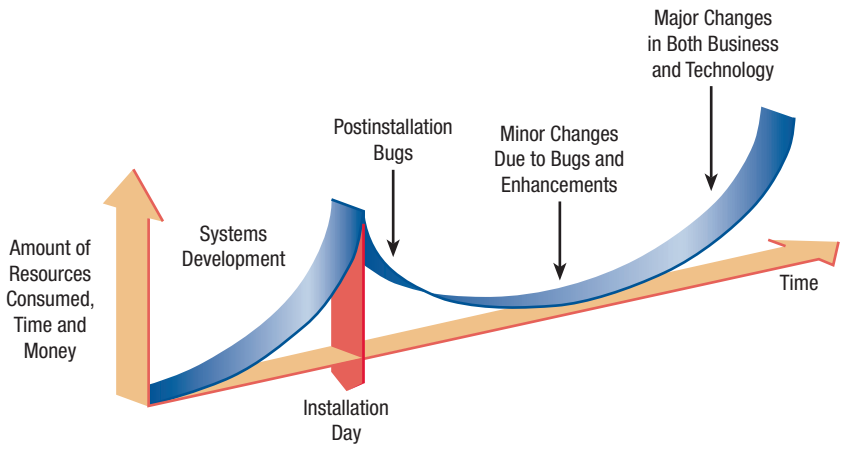
1. Users often request additional features after they become familiar with the computer system and its capabilities.
2. The business changes over time.
3. Hardware and software are changing at an accelerated pace.

Figure 1.2 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 the cost 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 software 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

FIGURE 1.2

Resource consumption over the system life.



system with a few minor enhancements. As time goes on and the business and technology change, however, the maintenance effort increases dramatically.

Using CASE Tools

Analysts who adopt the SDLC approach often benefit from productivity tools, called computer-aided software engineering (CASE) tools, 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.

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 1.3). Analysis reports may be produced using the repository information to show where the design is incomplete or contains errors.

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 software products such as Microsoft Visio or OmniGraffle

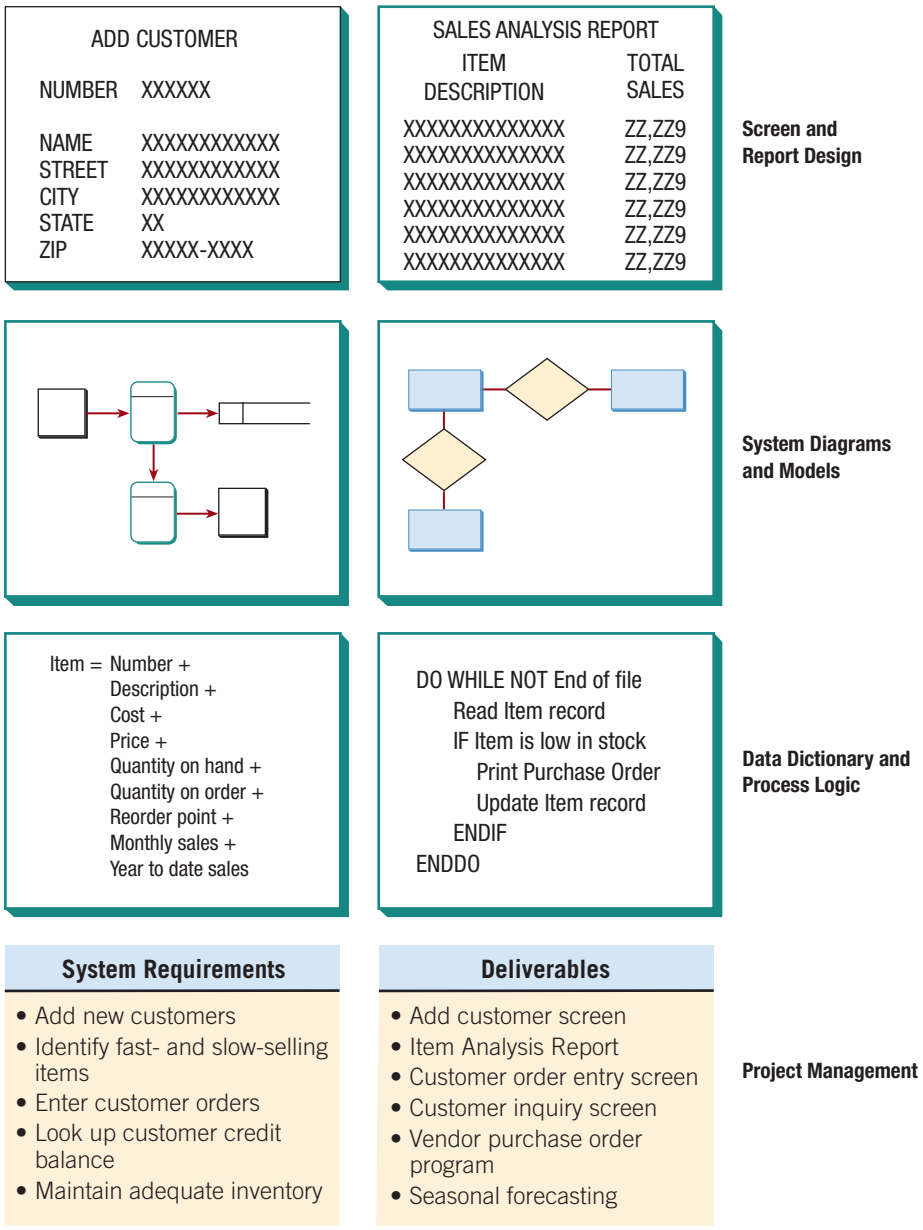


FIGURE 1.3
The repository concept.

allow users to draw and modify diagrams easily. However, VA is a full-fledged CASE tool with a repository and other features, whereas the other two are not.

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. CASE tools 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.

The Agile Approach

Although this text tends to focus on SDLC, the most widely used approach in practice, at times an analyst will recognize that his or her organization could benefit from an alternative approach. Perhaps a systems project using a structured approach has recently failed, or perhaps the organization 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 a 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 a 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.

We also explore an agile method called Scrum, which is named after a starting position in the sport of rugby. Scrum's success has much to do with its extremely quick releases. A typical sprint cycle will last between two and four weeks. At the end of that period, the team is expected to produce a potentially releasable product. That means that applications or websites are constantly changing. Each iteration boasts a new set of features produced during the sprint cycle. Scrum is also unique because team members choose what they want to work on as a team. Scrum is discussed further in Chapter 6. Let's return to discussing the agile approach in general.

Activities and behaviors 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*. Figure 1.4 illustrates the five distinct stages of the agile approach: exploration, planning, iterations to the first release, productionizing, and maintenance. Notice the three red arrows that loop back into the "Iterations" box and symbolize incremental changes created through repeated testing and feedback that eventually lead to a stable but evolving system. Also note that multiple looping arrows 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, illustrating a continuous feedback loop involving customers and the development team as they agree to alter the evolving system.

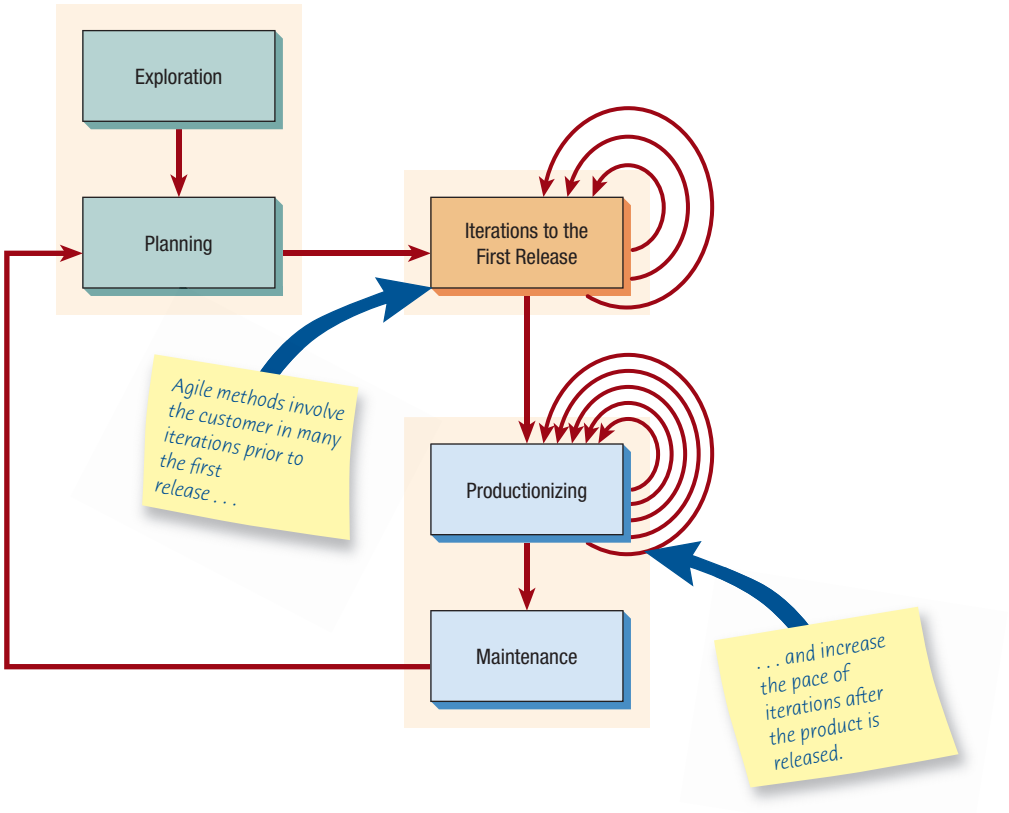


FIGURE 1.4

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

Exploration

During exploration, you 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 takes anywhere from a few weeks (if you already know 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 goal 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 take only 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 Kent Beck, the father of Extreme Programming. 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.

Using the metaphor of a game, 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. To arrive at the value, you have to deduct costs of development, and the time, expense, and uncertainty taken on so that the development project can 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.

The two main players in the planning game are 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 iterations (cycles of testing, feedback, and change) are 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 function 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 the productionizing 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 it 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 keys to the agile approach, which we heartily embrace, is that it is supposed to be fun to develop systems!

Maintenance

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

Object-oriented techniques often 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 O-O systems, called unified modeling language (UML), to break down a system into a use case model.

Object-oriented programming differs from traditional procedural programming in that it examines 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.

Object-Oriented Similarities to SDLC

The phases in UML are similar to those in the SDLC. Because these two methods share rigid and exacting modeling, they happen at a slower, more deliberate pace than the phases of agile modeling. An analyst goes through problem and identification phases, an analysis phase, and a design phase, as shown in Figure 1.5.

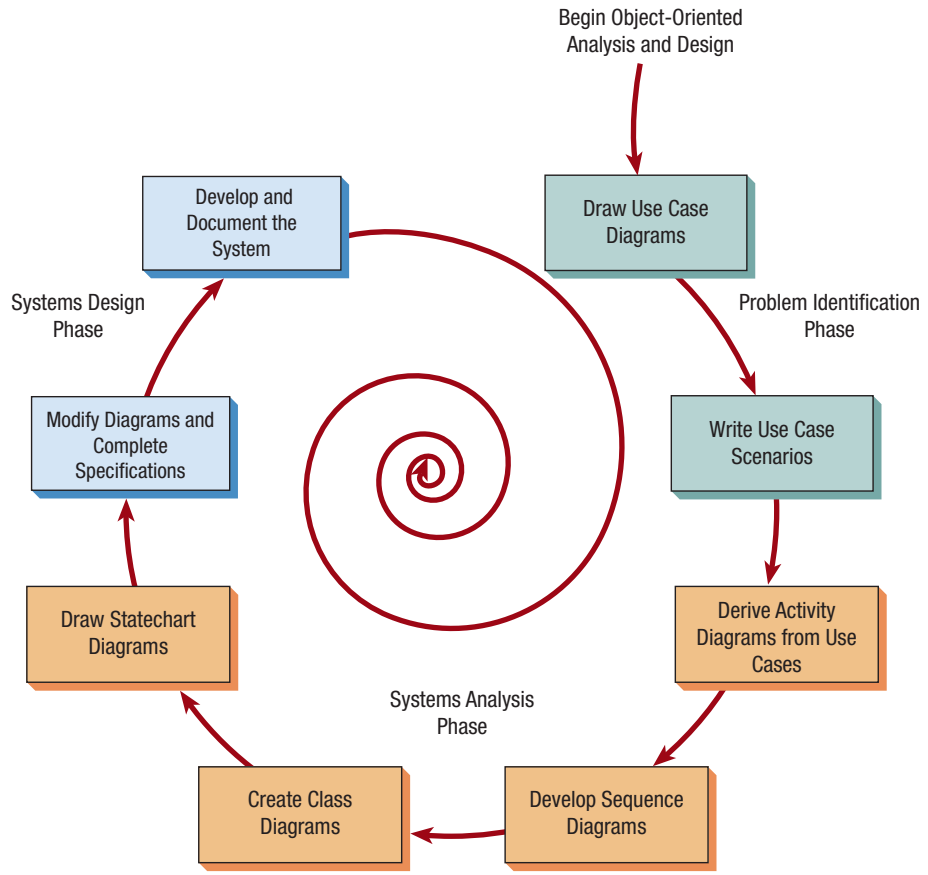


FIGURE 1.5

The steps in the UML development process.

Although the specifics are discussed in Chapters 2 and 10, the following steps provide a brief description of the UML process.

DEFINE THE USE CASE MODEL In this phase, the analyst identifies the actors and the major events initiated by the actors. Often the analyst starts 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.

DURING THE SYSTEMS ANALYSIS PHASE, BEGIN DRAWING UML DIAGRAMS In the second phase (Chapter 10), the analyst draws activity diagrams, which illustrate all the major activities in the use case. In addition, the analyst creates one or more sequence diagrams for each use case that 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.

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.

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.

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). An analyst will need to write class specifications for each class, including the attributes, methods, and their descriptions. The analyst also will develop method

specifications that detail the input and output requirements for each method, along with a detailed description of the internal processing of the method.

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 to the development team through documentation and UML diagrams is, 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 with a high-priority item or perhaps the item 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 this is repeated until the project is complete. 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 here are not as large 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 time and resources and develop a project proposal (Chapter 3). Next, they need to interview organization members and gather detailed data using questionnaires (Chapter 4) and sample data from existing reports and by observing 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 OOAs both require extensive planning and diagramming. The agile approach and the OOA both allow subsystems to be built one at a time until the entire system is complete. The agile and SDLC approaches are both concerned with 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 1.6 provides a set of guidelines to help you choose which method to use when developing your next system.

FIGURE 1.6

How to decide which development method to use.

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

Developing 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, many users and coders can study, share, and modify the code, or computer instructions. Rules of this community include the idea that any program modifications must be shared with all the people on the project and that all licenses must be adhered to.

Development of OSS also has been characterized as a philosophy rather than simply as the process of creating new software. Those involved in OSS communities often 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 open source groups need to understand where they converge and where the potential for conflict exists.

Why Organizations Participate in Open Source Communities?

Organizations participate in open source communities for a variety of reasons. One is the rapidity with which new software can be developed and tested. It is faster to have a committed group of expert developers develop, test, and debug code than it is to have one isolated team working on software development. This cross-fertilization can be a boon to creativity.

Another reason to participate is the benefit of having many good minds working on innovative applications. Yet another reason for participating in an open source community might be the potential for keeping down development costs.

Organizations might seek to participate in an open source community due to a desire to bolster their own self-image and to contribute something worthwhile to the larger software development community. They may want to contribute generously or altruistically to a higher good beyond developing profitable proprietary software, and doing so may make them appear as "good guys" to the external public.

Organizations may ask or even require that their software developers become involved in one or more open source projects or communities, but individual developers must interact with the community in a meaningful and knowledgeable way first to prove themselves worthy members of the group based on their merits and then to strike up and maintain relationships that are mutually beneficial.

Organizations and the design teams within them interact with Open Communities. Companies are taking advantage of an entire range of options that help them strike a harmonious equilibrium so that their contributions to the open community and their differentiation from the open community become clear strategically. Reasons for contribution to and differentiation from include cost, managing resources, and the time it takes to bring a new product to the market.

The Role of the Analyst in Open Source Software

As an analyst you may find yourself, at the request of your chief employer, participating in an open source community. One widely known open source community is that surrounding the Linux kernel. This is a large, mostly virtual community of developers who all have different levels or types of participation and who all have different reasons for being involved. Other well-known open source projects include Mozilla Firefox, Android, Apache projects, and many more. Even NASA, the U.S. National Aeronautics and Space Administration, has a lively open source community (see http://ti.arc.nasa.gov/opensource/).

One reason your company may ask you to participate in an open source community is curiosity about what the software benefits to the organization might be. This may be a result of a sort of bandwagon effect, for when it becomes known that competitors are already

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 Mr. 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 website www.pearsonglobaleditions.com. 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 website contains useful information about the project as well as files that can be downloaded to your computer. There is a set of Visible Analyst (VA) data files, and there is 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 website 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.

participating, your organization may want to get involved. With competitors actively participating in an open community, an organization may calculate that it is something that should at least be investigated seriously, not dismissed summarily. Another reason your company might ask you to participate as a developer in an open community is to achieve what researchers have labeled "responsive design." *Responsive design* means that while you are participating in the open source community, you are at the same time employed by an organization that wants to leverage your participation in the open source community to incorporate OSS designs into proprietary products, processes, knowledge, and IT artifacts that it is developing and that it hopes to eventually sell as a product that is differentiated from what the open source community has produced. Through a process of *responsive design* the IT artifact is imbued with both community and organization structures, knowledge, and practices.

Summary

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 problems. 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. It is important that you design new systems with an awareness of what you can do to design security into a system from the very beginning.

Analysts possess a wide range of skills. First and foremost, an 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 enable 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. Scrum is an agile method that emphasizes short releases and allows teams to choose what they want to work on. 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 typical in object-oriented analysis and design.

Analysts will have increasing opportunities to participate in open source development communities, often through their primary organization. Organizations participate in development of open source for many reasons. One of the well-known open source communities maintains the Linux kernel and is supported by the Linux Foundation.

Keywords and Phrases

agent of change
agile approach
agile methods
Ajax
bespoke software
computer-assisted software engineering (CASE)
CASE tools
exploration phase
human–computer interaction (HCI)
iterations to the first release phase
Linux kernel
maintenance phase
object-oriented systems analysis and design
open source communities
open source software (OSS)
planning game
planning phase
productionizing phase
prototyping
Scrum
security
systems analysis and design
systems analyst
systems consultant
systems development life cycle (SDLC)
unified modeling language (UML)

Review Questions

1. Why does a lack of proper training often result in the failure of a new system?
2. Why has the importance of the user in systems development changed?
3. List three roles that a systems analyst is called upon to play. Provide a definition for each one.
4. List the three primary roles of the systems analyst.
5. List and briefly define the seven phases of the systems development life cycle (SDLC).
6. Why is human–computer interaction (HCI) important for systems analysis?
7. What are the four values of the *agile approach*?
8. What is the meaning of the phrase *the planning game*?
9. Discuss the importance of the maintenance phase in designing new systems.
10. What is Scrum?
11. Describe a situation in which an analyst would use object-oriented systems analysis and design rather than the systems development life cycle.
12. What are class diagrams in the object-oriented approach (OOA)?
13. What are the conditions for choosing a systems development method?
14. What is the role of a systems analyst in the iterations and the productionizing phase of the agile approach?
15. List two reasons an organization may want its analysts to participate in an open source community.

Selected Bibliography

Beck, K., and C. Andres. *Extreme Programming Explained: Embrace Change,* 2d ed. Boston, MA: Addison-Wesley, 2004.

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, NY: McGraw-Hill, 1985.

Germonprez, M., and B. Warner. "Commercial Participation in Open Innovation Communities." In *Managing Open Innovation Technologies*, ed. E. Lundström, J. S. Z, M. Wiberg, S. Hrastinski, M. Edenius, and P. J. Ägerfalk. Berlin: Springer-Verlag, 2012.

Germonprez, M., J. E. Kendall, K. E. Kendall, L. Mathiassen, B. Young, and B. Warner. "A Theory of Responsive Design: A Field Study of Corporate Engagement with Open Source Communities." *Information Systems Research* 28, 1 (2017): 64–83.

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*, 201–222, ed. E. W. Duggan and H. Reichgelt. Hershey, PA: Idea Group Publishing, 2006.

Kendall, K. E., S. Kong, and J. E. Kendall. "The Impact of Agile Methodologies on the Quality of Information Systems: Factors Shaping Strategic Adoption of Agile Practices." *International Journal of Strategic Decision Sciences* 1, 1 (2010): 41–56.

Laudon, K. C., and J. P. Laudon. *Management Information Systems,* 15th ed. Hoboken, NJ: Pearson, 2018.

Lee, G., and R. Cole. "From a Firm-Based to a Community-Based Model of Knowledge Creation: The Case of the Linux Kernel Development." *Organization Science* 14 (2003): 663–649.

Visible Enterprise Products. www.visible.com/Products/index.htm. Last accessed August 16, 2017.

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* 15 (2005): 512–543.