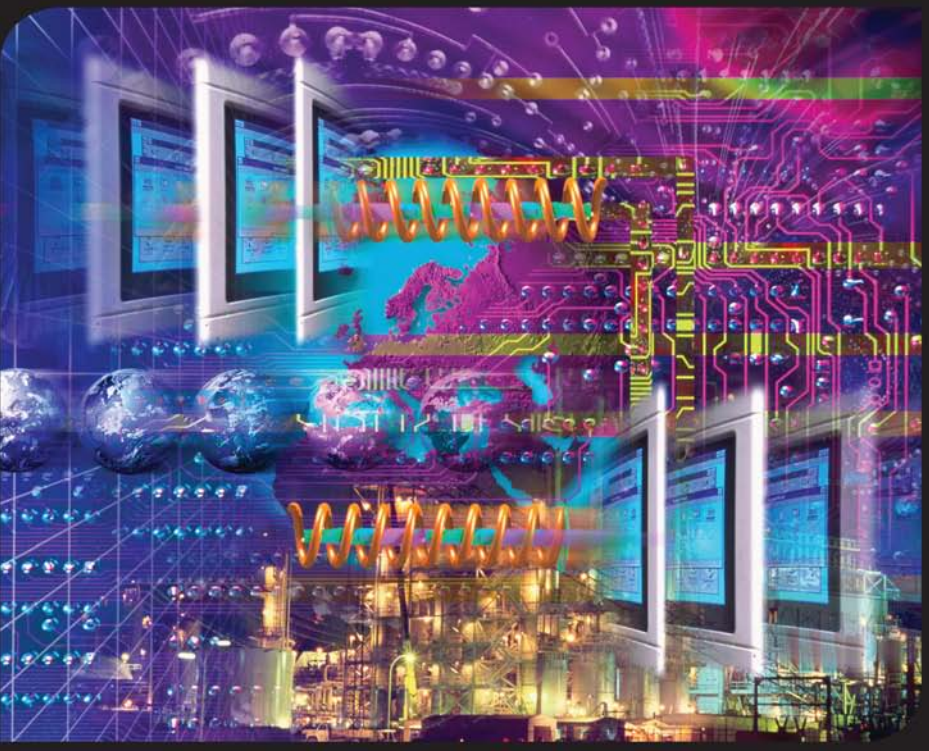
Systems Architecture

7e



Stephen D. Burd

SYSTEMS ARCHITECTURE

Seventh Edition

Stephen D. Burd
University of New Mexico

CENGAGE Learning®

Australia • Brazil • Mexico • Singapore • United Kingdom • United States

Systems Architecture, Seventh Edition
Stephen D. Burd

Vice President, General Manager, Science, Math & Quantitative Business: Balraj Kalsi

Product Director: Joe Sabatino

Product Manager: Jason Guyler

Content Developer: Lori Bradshaw, S4 Carlisle

Senior Product Assistant: Brad Sullender

Senior Marketing Manager: Eric La Scola

Marketing Coordinator: William Guiliani

Art and Cover Direction, Production Management, and Composition: Lumina Datamatics, Inc.

Intellectual Property

Analyst: Christina Ciaramella

Project Manager: Kathryn Kucharek

Manufacturing Planner: Ron Montgomery

Cover Image: Oliver Burston (Debut Art)/ The Image Bank/Getty Images

Library of Congress Control Number: 2015938232

ISBN: 978-1-305-08019-5

Cengage Learning
20 Channel Center Street
Boston, MA 02210
USA

Cengage Learning is a leading provider of customized learning solutions with employees residing in nearly 40 different countries and sales in more than 125 countries around the world. Find your local representative at **www.cengage.com**

Cengage Learning products are represented in Canada by Nelson Education, Ltd.

To learn more about Cengage Learning Solutions, visit **www.cengage.com**

Purchase any of our products at your local college store or at our preferred online store **www.cengagebrain.com**

Printed in the United States of America
Print Number: 01 Print Year: 2015

To Alison Colton and Miles Fisher—thanks for many years of music and friendship.

CONTENTS

PREFACE

INTENDED AUDIENCE

This book is intended for undergraduate students majoring or concentrating in information systems (IS) or information technology (IT) and as a reference for IS/IT professionals. It provides a technical foundation for systems design, systems implementation, hardware and software procurement, and computing resource management. Computer hardware and system software topics that are most useful to IS/IT students and professionals are described at an appropriate level of detail. For some topics, readers gain enough knowledge to solve technical problems. For other topics, they learn what they need to communicate effectively with technical specialists.

Computer science students are exposed to computer hardware and system software technology in many undergraduate courses. Computer science books usually focus on a subset of the topics in this book. However, coverage of hardware and system software in an IS/IT curriculum is usually limited. A brief overview of hardware and system software might be provided in an introductory course, and some specific technical topics are often covered in other courses, but there's at most one course devoted to hardware and system software.

At this writing (March 2015), the latest curricula recommendations in IS and IT are IS 2010 and IT 2008. Some schools are still using curricula modeled on IS 2002 (see *www.acm.org* for details on these curricula). The topics covered in this book are mapped to all three curricula as follows:

- *IS 2002*—This book covers a superset of the requirements for IS 2002.4, Information Technology Hardware and System Software. Additional topics beyond those in IS 2002.4 include networks, application development software, and system administration.
- *IT 2008*—This book covers topics in four of the body of knowledge components: Integrative Programming and Technologies—Intersystems Communications and Overview of Programming Languages; Networking—all topics; Platform Technologies—all topics except Enterprise Deployment Software; and Systems Administration and Maintenance—Operating Systems and portions of Applications and Administrative Activities.
- *IS 2010*—This book covers the topics and learning objectives of the IS 2010.4 core course, IT Infrastructure. It also covers a subset of the topics and learning objectives of the IS 2010.3 core course, enterprise architecture.

This book can also serve as a supplement in courses on system design and computer resource management. For system design, it covers many technical topics to address when selecting and configuring hardware and system software. For computer resource management, it offers the broad technical foundation needed to manage resources effectively.

READERS' BACKGROUND KNOWLEDGE

Because target courses for this book are typically placed early in the recommended curricula, few assumptions are made about readers' background knowledge. Unlike many computer science books, readers aren't assumed to have an extensive background in mathematics, physics, or engineering. When necessary, background information in these areas is provided in suitable depth.

In addition, readers aren't assumed to know any particular programming language. However, classroom or practical experience with at least one language is helpful to comprehend the discussions of CPU instruction sets, operating systems, and application development software. Programming examples are given in several programming languages and in pseudocode.

Detailed knowledge of a particular operating system isn't required. However, as with programming experience, practical experience with at least one operating system is helpful. Lengthy examples from operating systems are purposely avoided, but there are some short examples from MS-DOS, UNIX/Linux, and recent Windows versions.

Finally, knowledge of low-level machine instructions or assembly-language programming isn't assumed. Assembly-language instructions are described in several chapters, but a generic assembly language is used, and no detailed coverage of assembly-language program organization is included.

CHANGES IN THIS EDITION

The sixth edition was first published in 2010. Updates were needed throughout the book to address changes since that time. The following sections summarize major updates and additions, although most chapters include many additional minor changes, such as updates to screen captures, hardware specifications, and standards.

- *Chapter 1*—Moved the definition of systems architecture and related terms to this chapter from Chapter 2, incorporated a Web-based application into the discussion of architecture and construction, updated the discussion of periodical literature and online sources of technology information, and removed material covering job titles and definitions.
- *Chapter 2*—Updated typical computer specifications and specifically incorporated a wider variety of portable and embedded computing devices, revised definitions of computer classes, updated the quantum computing Tech Focus to incorporate commercially available quantum computers, expanded the introduction to the Internet and computer networks, incorporated Web-based

applications, Web services, and embedded software into the software introduction, moved some operating system details to Chapter 11, updated the Business Focus case, and updated the Technology Focus features on IBM POWER processors and the parallel evolution of Intel CPUs and Microsoft operating systems.

- *Chapter 3*—Minor updates.
- *Chapter 4*—Scaled back the discussions of instruction formats and RISC vs. CISC, updated the discussion of RISC and the Pentium Technology Focus, updated the Technology Focus on SPEC and TPC benchmarks, and updated several sections to reflect current CPU clock rates, word sizes, fabrication technology, and multicore architecture.
- *Chapter 5*—Updated and expanded coverage of solid-state drives including a new Tech Focus on server-oriented SSDs, updated the coverage of memory packaging and nonvolatile memory technologies, scaled back the coverage of magnetic tape, added coverage of cloud-based storage, and modernized device specifications throughout the chapter.
- *Chapter 6*—Modernized the coverage of bar codes, added coverage of touch-screen technology, and modernized device specifications throughout the chapter.
- *Chapter 7*—Reduced the coverage of older display technologies and expanded the discussion of current display types and video adapters.
- *Chapter 8*—Moved specifics of wireless networking standards to Chapter 9, expanded and clarified the discussion of bandwidth, digital signals, and data transfer rates, and simplified the discussion of optical cabling, updated the coverage of parallel and serial transmission including the related Tech Focus, added a discussion of modern line coding methods, refined the CRC coverage, updated the InfiniBand Technology Focus to also cover Fibre Channel, and updated the Business Focus.
- *Chapter 9*—Expanded Ethernet coverage include updates to latest and emerging standards, expanded and modernized coverage of wireless networking standards, and updated the WiMax Tech Focus and Business Focus.
- *Chapter 10*—Minor updates.
- *Chapter 11*—Incorporated O/S overview material moved from Chapter 2, updated VMware Tech Focus, updated figures supporting the memory allocation discussion, and updated device driver discussion including supporting figures.
- *Chapter 12*—Reduced emphasis on magnetic storage and increased emphasis on SSDs, updated Windows screen captures throughout the chapter, added a new section on cloud-based storage, and updated Google File System Tech Focus.
- *Chapter 13*—Updated coverage of the Internet and Web standards, added material covering Web-based applications and services, and modernized the discussion of connection standards and infrastructure.
- *Chapter 14*—Minor updates.

RESOURCES FOR INSTRUCTORS

Systems Architecture, Seventh Edition includes the following resources to support instructors in the classroom. All the teaching tools available with this book are provided to the instructor on a single CD. They can also be accessed with your single sign-on (SSO) account at Cengage.com.

- *Instructor's Manual*—The Instructor's Manual provides materials to help instructors make their classes informative and interesting. It includes teaching tips, discussion topics, and solutions to end-of-chapter materials.
- *Classroom presentations*—Microsoft PowerPoint presentations are available for each chapter to assist instructors in classroom lectures or to make available to students.

Cengage Learning Testing Powered by Cognero is a flexible, online system that allows you to:

- author, edit, and manage test bank content from multiple Cengage Learning solutions
- create multiple test versions in an instant
- deliver tests from your LMS, your classroom or wherever you want

- *Distance learning content*—Cengage Learning is proud to present online content in WebCT and Blackboard to provide the most complete and dynamic learning experience possible. For more information on how to bring distance learning to your course, contact your local Cengage sales representative.

WORLD WIDE WEB SITES

Two support sites for this book (instructor and student), located at *www.cengage.com/mis/burd*, offer the following:

- The Instructor's Manual
- Figure files
- End-of-chapter questions and answers
- Web resource links for most book topics and research problems
- Additional online chapter on virtualization
- Text updates and errata
- Glossary

ORGANIZATION

This book's chapters are organized into four groups. The first group contains two chapters with overviews of computer hardware, software, and networks and describes sources of technology information. The second group consists of five chapters covering hardware technology. The third group contains two chapters on data communication and computer networks. The fourth group includes five printed chapters covering software technology and system administration and an online chapter covering virtualization.

The chapters are intended for sequential coverage, although other orderings are possible. The prerequisites for each chapter are described in the following section. Other chapter orders can be constructed based on these prerequisites. Chapter 2 should always be covered before other chapters, although some sections can be skipped without loss of continuity, depending on which subsequent chapters are included or skipped.

There should be time to cover between 9 and 12 chapters in a three-credit-hour undergraduate course. This book contains 14 chapters to offer flexibility in course content. Topics in some chapters can be covered in other courses in a specific curriculum. For example, Chapters 8 and 9 are often covered in a separate networking course, and Chapter 14 is often included in a separate system administration course. Instructors can choose specific chapters to best match the overall curriculum design and teaching preferences.

CHAPTER DESCRIPTIONS

Chapter 1, "Computer Technology: Your Need to Know," briefly describes how knowledge of computer technology is used in the systems development life cycle. It also covers sources for hardware and system software information and lists recommended periodicals and Web sites. It can be skipped entirely or assigned only as background reading.

Chapter 2, "Introduction to Systems Architecture," provides an overview of hardware, system and application software, and networks. It describes main classes of hardware components and computer systems and describes the differences between application and system software. This chapter introduces many key terms and concepts used throughout the book.

Chapter 3, "Data Representation," describes primitive CPU data types and common coding methods for each type. Binary, octal, and hexadecimal numbering systems and common data structures are also discussed. Chapter 2 is a recommended prerequisite.

Chapter 4, "Processor Technology and Architecture," covers CPU architecture and operation, including instruction sets and assembly-language programming. It describes traditional architectural features, including fetch and execution cycles, instruction formats, clock rate, registers, and word size. It also discusses methods for enhancing processor performance as well as semiconductor and microprocessor fabrication technology. Chapters 2 and 3 are necessary prerequisites.

Chapter 5, "Data Storage Technology," describes implementing primary and secondary storage with semiconductor, magnetic, and optical technologies. Principles of each storage technology are described first, followed by factors affecting each technology and guidelines for choosing secondary storage technologies. Chapters 3 and 4 are necessary prerequisites, and Chapter 2 is a recommended prerequisite.

Chapter 6, "System Integration and Performance," explains communication between computer components and performance enhancement methods. It starts with a discussion of system bus and subsidiary bus protocols, followed by coverage of device controllers, mainframe channels, and interrupt processing. Performance enhancement methods include buffering, caching, parallel and multiprocessing, and compression.

Chapters 4 and 5 are required prerequisites, and Chapters 2 and 3 are recommended prerequisites.

Chapter 7, "Input/Output Technology," describes I/O devices, including keyboards, pointing devices, printers and plotters, video controllers and monitors, optical input devices, and audio I/O devices. It also covers fonts, image and color representation, and image description languages. Chapter 3 is a necessary prerequisite, and Chapters 2, 5, and 6 are recommended prerequisites.

Chapter 8, "Data and Network Communication Technology," covers data communication technology, beginning with communication protocols, analog and digital signals, transmission media, and bit-encoding methods. This chapter also explains serial and parallel transmission, synchronous and asynchronous transmission, wired and wireless transmission, channel-sharing methods, clock synchronization, and error detection and correction. Chapters 2 and 3 are recommended prerequisites.

Chapter 9, "Computer Networks," describes network architecture and hardware. It starts with network topology and message forwarding and then explains media access control and network hardware devices, such as routers and switches. This chapter also covers IEEE and OSI networking standards and includes an in-depth look at Internet architecture and TCP/IP. Chapters 3, 4, and 8 are necessary prerequisites, and Chapter 2 is a recommended prerequisite.

Chapter 10, "Application Development," begins with a brief overview of the application development process and development methodologies and tools, and then discusses programming languages, compilation, link editing, interpretation, and symbolic debugging. The final section describes application development tools, including CASE tools and integrated development environments (IDEs). Chapters 2, 3, and 4 are necessary prerequisites.

Chapter 11, "Operating Systems," describes the functions and layers of an operating system, explains resource allocation, and describes how an operating system manages the CPU, processes, threads, and memory. Chapters 2, 4, and 5 are necessary prerequisites, and Chapter 10 is a recommended prerequisite.

Chapter 12, "Secondary Storage Management," gives an overview of file management components and functions, including differences between logical and physical secondary storage access, and describes file content and structure and directories. Next, this chapter describes storage allocation, file manipulation, and access controls and ends with file migration, backup, recovery, fault tolerance, and storage consolidation methods including cloud-based storage. Chapters 5 and 11 are necessary prerequisites, and Chapter 10 is a recommended prerequisite.

Chapter 13, "Internet and Distributed Application Services," begins by discussing distributed computing and network resource access, network protocol stacks, and directory services. This chapter also explains interprocess communication, Internet protocols for accessing distributed resources, and component-based application development. Finally, it describes Web-based applications, Web services, and cloud computing models. Chapters 2, 8, and 9 are necessary prerequisites. Chapters 4, 11, and 12 are recommended prerequisites.

Chapter 14, "System Administration," gives an overview of system administration tasks and the strategic role of hardware and software resources in an organization. It then describes the hardware and software acquisition process. Next, the chapter discusses methods for determining requirements and monitoring performance. The next section describes measures for ensuring system security, including access controls, auditing, virus protection, software updates, and firewalls. The last section discusses physical environment factors affecting computer operation. Chapters 2, 4, 8, 11, and 12 are recommended prerequisites.

Online Chapter, "Virtualization," provides in-depth coverage virtualization topics, including history, virtualization layers, hardware support, networking issues, the benefits, challenges, and risks of virtualization, and typical deployment scenarios.

Appendix, "Measurement Units," summarizes common measurement units, abbreviations, and usage conventions for time intervals, data storage capacities, and data transfer rates.

ACKNOWLEDGMENTS

The first edition of this book was a revision of another book, *Systems Architecture: Software and Hardware Concepts*, by Leigh and Ali. Some of their original work has endured through all subsequent editions. I am indebted to Leigh, Ali, and Cengage Learning for providing the starting point for all editions of this book.

I thank everyone who contributed to this edition and helped make previous editions a success. Jim Edwards took a chance on me as an untested author for the first edition. Kristen Duerr, Jennifer Locke, Maureen Martin, and Kate Mason shepherded the text through later editions. Jason Guyler oversaw this edition. Thanks to all past and present development and production team members.

I thank students in the undergraduate MIS concentration at the Anderson Schools of Management, University of New Mexico, who have used manuscripts and editions of this book over the past two and a half decades. Student comments have contributed significantly to improving the text. I also thank my department chair, Steven Yourstone, and my faculty colleagues—Ranjit Bose, William Bullers, Nick Flor, Aaron French, Peter Jurkat, Xin Luo, Laurie Schatzberg, Josh Saiz, and Alex Seazzu—for their continued support and encouragement of my textbook-writing activities.

Finally, I'd like to thank Dee, my wife, and Alex and Amelia, my children. Developing this book as a sole author through multiple editions has been a time-consuming process that often impinged on family time and activities. My success as an author would not be possible without my family's love and support.

CHAPTER 1

COMPUTER TECHNOLOGY: YOUR NEED TO KNOW

LEARNING OBJECTIVES

After studying this chapter, you should be able to:

- Define systems architecture and related terms
- Explain the relationship between architecture, design, and technology
- Describe what technology knowledge is required to develop information systems and manage computing resources
- Describe sources of architecture- and technology-related knowledge and how to effectively use each

OVERVIEW

The words you're reading now result, in part, from computer-based information systems. The author, editors, production team, and distribution team all relied on computer systems to organize, produce, and then convey this information. Although many different kinds of computer information systems were involved, they all share similar technology: Each consists of computer hardware, software, data, and communication capabilities. In this chapter, you learn why you need to study systems architecture and computer-related technology if you work, or plan to work, in information systems. You also learn about additional sources of information that can help expand and update your knowledge of computer-related technology.

WHAT IS SYSTEMS ARCHITECTURE?

The term *architecture* is usually associated with buildings but is often applied in other contexts, such as landscapes, computers, and information systems. Definitions of the term *architecture* typically include words like structure, organization, and integration. The architecture of a complex structure such as a building or computer system describes its component pieces and how they're organized or integrated to create the entire structure. This view tends to emphasize the static nature of the components though their dynamic interaction is also considered, such as when a building architect models the flow of people through a hallway or the impact of wind or water on a bridge.

Computer systems are complex combinations of hardware, software, and network components. Information systems include computer systems, the people that interact with them, and the data and information that are processed and communicated. Computer and information systems contain many interacting *parts* organized by a well-defined structure. Therefore, the term *architecture* and the processes of architecture and design are as applicable to computers and information systems as they are to buildings, bridges, and hydroelectric dams.

The term **systems architecture** describes the structure, interaction, and technology of computer or information system components. Because the term *systems architecture* is very general, there are many other terms that more specifically incorporate the term *architecture* in the context of computers and information systems. Examples include the following:

- ***Computer architecture***—Architecture of a single computer system or a group of cooperating computer systems.
- ***Information architecture***—Architecture of data or information—for example, a database schema or the structure of a document archive.
- ***Network architecture***—Architecture of a computer network including wired and wireless connections, network-specific hardware such as routers, and related communication protocols and software.
- ***Software architecture***—Architecture of a program, software subsystem, or software system.
- ***Technology architecture***—Combination of all four previously mentioned terms with specific emphasis on issues such as performance, reliability, compatibility, and extensibility.

Architecture, Design, and Technology

Architecture is the process of analysis and design. A building architect first analyzes user requirements for a building, such as size and function, and constraints, such as the building site and available budget. A building architect creates drawings that depict only a building's appearance and internal layout from multiple perspectives. Early drawings are simple. As design decisions are made, diagrams are redrawn with increasing amounts of detail. As with building architecture, systems architecture is often summarized by diagrams. Figure 1.1 shows an example of a network diagram that summarizes the computer hardware and network connections that support an online shopping system.

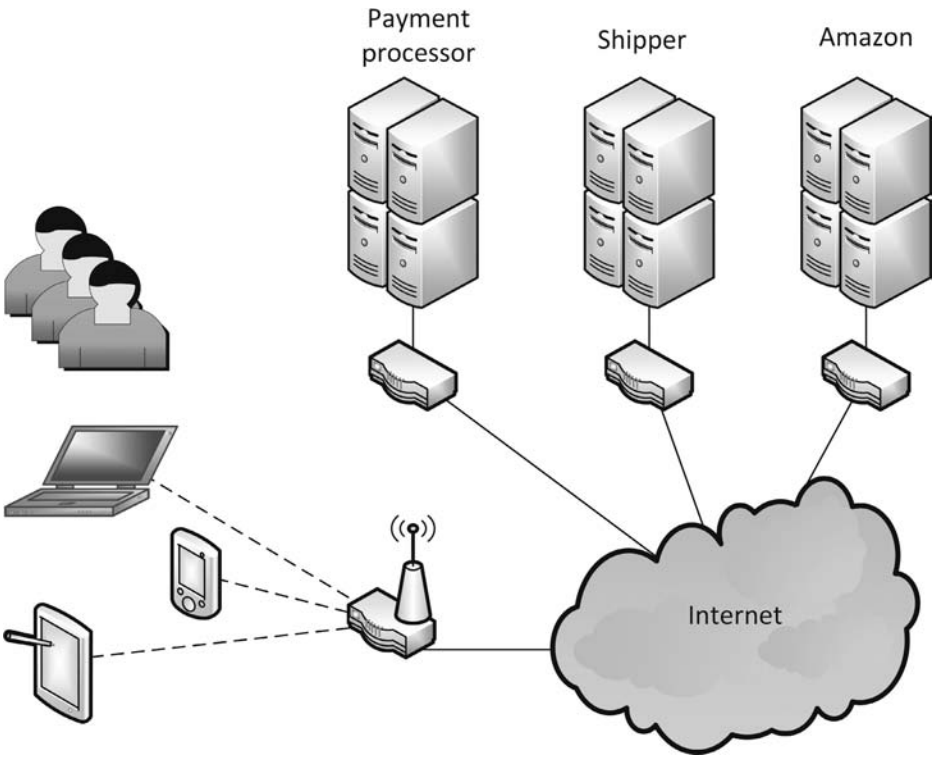


FIGURE 1.1 Network diagram of an online shopping system

With both buildings and systems, architects must understand the underlying technology of what they design. When studying building architecture, an architecture student learns about many technology-related topics such as the strength and durability of various building materials, structural engineering principles, and construction methods. Though architects don't usually build the buildings they design, their designs and detailed construction plans tell others how to do so. An architect needs a broad and deep knowledge of building and construction technology to design buildings that are durable and can be constructed on time and within budget.

The relationship between architecture, design, and technology also applies to computers, software, networks, and information systems. For example, designing an operating system requires detailed knowledge of the underlying computer hardware. Designing an app that interacts with server-side software components requires detailed knowledge of operating systems, network protocols, and programming toolkits. Designing a technology architecture that will effectively and efficiently support all of an organization's systems now and in the future requires technical knowledge that spans a wide range of hardware, software, network, and information technology.

Though the title of this book—*Systems Architecture*—implies a focus on architecture and design, the primary focus of the book is technology. What you learn about computer technology in this book provides a foundation for design and architecture topics covered briefly in this text and in much greater depth in other courses you will take.

WHO NEEDS TO UNDERSTAND SYSTEMS ARCHITECTURE?

The world is filled with complex technical devices that ordinary people use every day. Fortunately, you don't need a detailed understanding of how these devices work to use them. Imagine needing three months of training just to use a refrigerator or needing a detailed understanding of mechanics and electronics to drive a car. Using the earliest computers in the 1940s took years of training, but today, even though computers are increasingly complex and powerful, they're also easier to use. If computers and information systems have become so easy to use, why do you need to know anything about their architecture and technology?

The knowledge required to purchase and configure technically complex devices is far greater than the knowledge required to use them effectively. Many people can use complex devices, such as cars, home theater systems, and computers, but few people feel comfortable purchasing or configuring them. Why is this so?

When you walk into a store or visit a Web site to purchase a computer, you're confronted with a wide range of choices, including processor type and speed, hard disk speed and capacity, memory capacity, and operating system. To make an informed choice, you must know your preferences and requirements, such as the application software you plan to use and whether you plan to discard or upgrade the computer in a year or two. To evaluate the alternatives and determine their compatibility with your preferences and requirements, you must be able to comprehend technical terms (for example, gigahertz, gigabyte, DDR, and USB), technical and sales documentation, product and technology reviews, and the advice of friends, experts, and salespeople.

An information systems (IS) professional faces computer acquisition, upgrade, and configuration choices that are far more complex. Large computer systems and the software that runs on them use more complex technology than smaller ones do. There are many more components and, therefore, more complex configuration, compatibility, and administrative issues. Of course, the stakes are higher. Employers and users rely on the expertise of IS professionals, and companies invest substantial sums of money based on their recommendations. Are you (or will you be) able to meet the challenge?

Information System Development

When developing an information system, IS professionals follow a series of steps called a **systems development life cycle (SDLC)**. Figure 1.2 shows a modern SDLC called the **Unified Process (UP)**. Under the UP, an information system is built in a series of four- to six-week repeated steps called *iterations* (the vertical columns separated by dashed outlines). Although Figure 1.2 shows six iterations, the number of iterations is tailored to each development project's specifications. Typically, the first iteration or two produces documentation and a prototype (model) system that's refined and expanded in subsequent iterations until it becomes the final system.

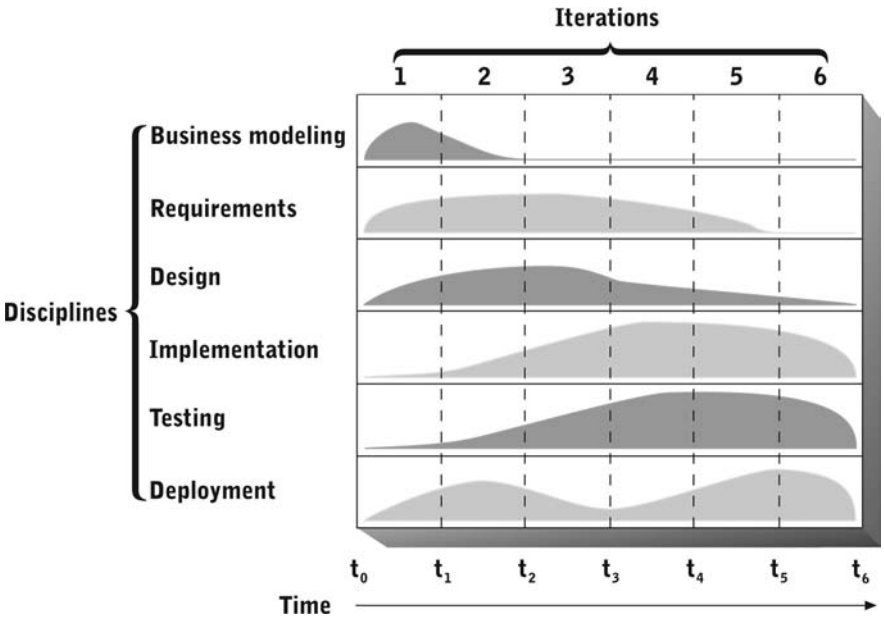


FIGURE 1.2 Disciplines and iterations in the Unified Process

Each iteration includes whatever activities are needed to produce testable models or working software. Related activities are grouped into UP **disciplines**. For example, the testing discipline includes activities such as creating test data, conducting tests, and evaluating test results. Activities and efforts in each discipline vary across iterations, as shown by the shaded curves in Figure 1.2. For example, in this figure, activities in iteration 1 are drawn primarily from the business modeling, requirements, design, and deployment disciplines, and activities in iteration 6 are drawn primarily from the implementation, testing, and deployment disciplines. As with the number of project iterations, the mix of activities in each iteration is tailored to each development project. Therefore, efforts in each discipline aren't always distributed across iterations exactly as shown in Figure 1.2.

The following sections explore the UP disciplines in more detail and describe the knowledge of systems architecture and technology each one requires.

Business Modeling and Requirements Disciplines

Activities of the **business modeling discipline** and the **requirements discipline** are primarily concerned with building models of the organization that will own and operate the system, models of the system's environment, and models of system and user requirements. The models can include narratives, organizational charts, workflow diagrams, network diagrams, class diagrams, and interaction diagrams. The purpose of building business and requirements models is to understand the environment in which the system will function and the tasks the system must perform or assist users to perform.

While building business and requirements models, developers ask many questions about the organization's needs, users, and other constituents and the extent to which

these needs are (or aren't) being met and how they'll be addressed by a new system. Technical knowledge of computer hardware and system software is required to assess the degree to which users' needs are being met and to estimate the resources required to address unmet needs.

For example, an analyst surveying an online shopping system might pose questions such as the following:

- How much time is required to find products and complete a sale?
- Is the system easy to use?
- Is enough information being gathered for related purposes such as marketing?
- Can the system handle periods of peak sales volume (such as during holidays)?
- Should the system be hosted locally or within a cloud computing environment?
- Is data transmitted between end-user devices and servers adequately protected?
- Will the current architecture adequately support future system upgrades?

Answering these questions requires an in-depth understanding of the underlying hardware and software technologies. For example, determining whether a system can respond to periods of peak demand requires detailed knowledge of processing and storage capabilities, operating systems, networks, and application software. Determining whether data in transit is adequately secured requires a detailed understanding of secure network protocols. Determining whether the current technology architecture unduly restricts future upgrades requires knowledge of the current architectural limitations and the features of emerging architectures and standards. Therefore, the architectural and technology knowledge required to perform business and requirements models covers a broad range of both present and future technology.

Design Discipline

The **design discipline** is the set of activities that determine the structure of a specific information system that fulfills the system requirements. The first set of design activities, called **architectural design**, selects and describes the exact configuration of all hardware, network, system software, and application development tools to support system development and operations (see Figure 1.3). These selections affect all other design decisions and serve as a blueprint for implementing other systems.

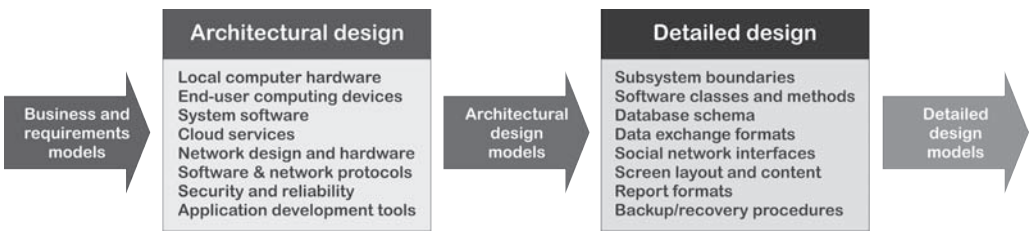


FIGURE 1.3 Design activities in the Unified Process

Specific systems design tasks include selecting and configuring the following:

- Computer hardware or cloud computing services
- Network hardware (transmission lines, routers, firewalls, and related protocols)
- System software (operating system, database management system, Web server software, network services, and security software and protocols)
- Application development tools (programming languages, component libraries, and integrated development environments)

These choices are combined to form a technology architecture within which application software and data components will be deployed. Technology components such as networks and servers typically support all of an organization's information systems. When a new system development project starts, the existing technology architecture is examined to determine whether it can support the new system. If not, the technology architecture is updated as needed though the updates must also support existing systems. Because the technology architecture covers many different technologies and supports many different systems, maintaining and updating that architecture is one of the most challenging tasks an IS professional must perform.

The remaining design activities, called **detailed design**, are narrower in scope and constrained by the technology architecture. Detailed design activities include the following:

- File or database design (such as grouping data elements into records and files, indexing, and sorting)
- Application software design
- User and external system interface design (input screen formats, report formats, and protocols to interact with external services and systems)
- Design of system backup and recovery mechanisms

Technical knowledge of computer hardware and system software is most important for performing architectural design activities. The knowledge required to evaluate and update technology architecture spans the full breadth of computer-related technology and a time dimension that includes older systems, newer systems, and systems that might be developed in the future. Specific architectural design subtasks often require deep specialized knowledge. For example, selecting hardware and network components requires detailed knowledge of their capabilities and limitations. Evaluating cloud-based services and deployment options also requires detailed knowledge of their capabilities and limitations. The designer must also consider their compatibility with existing internal hardware, network components, and system software.

Individual detailed design activities typically require less breadth in technology knowledge though greater depth in areas that support the specific task. For example, selecting appropriate development tools requires knowing the information system requirements and capabilities of the hardware, network, and operating system. Development tools (and the software components built with them) vary widely in their efficiency, power, and compatibility. Tool selection also affects future system development projects.

Implementation and Testing Disciplines

The **implementation discipline** of the UP includes all activities for building, acquiring, and integrating application software components. The **testing discipline** includes activities that verify correct functioning of infrastructure and application software components and ensure that they satisfy system requirements. Implementation and especially testing activities require specific knowledge of the hardware, network, and system software.

For example, developing an application software component that interacts with an external Web service to schedule a shipment requires specific knowledge of the network protocols used to find and interact with the service. Diagnosing an error that occurs when the software executes requires detailed knowledge of the operating system, network services, and network protocols for creating, transmitting, and receiving the Web service request and response.

Deployment Discipline

The **deployment discipline** is the set of activities required for installing and configuring infrastructure and application software components and bringing them into operation. Questions addressed by deployment discipline activities include the following:

- Who should be involved in and responsible for deploying each part of the system?
- In what order should parts of the system be deployed?
- Will any parts of the new system operate in parallel with the previous system?

Technical knowledge of computer hardware and system software is needed to perform many deployment tasks. Installing and configuring hardware, networks, and system software is a specialized task that requires a thorough understanding of the components being installed and the purposes for which they'll be used. Tasks such as formatting storage devices, setting up system security, installing and configuring network services, and establishing accounting and auditing controls require considerable technical expertise.

Evaluation and Maintenance

Although not a formal UP discipline, evaluation and maintenance are important activities that account for much of the long-range system cost. Over time, problems with the system can and do happen. Errors that escaped detection during testing and deployment might show up. For example, a Web-based order-entry system might become overloaded because of inadequate estimates of processing volume, network congestion, or capacity limits in underlying hardware or database services. Information needs can and do change, necessitating changes to collect, process, and store additional data.

Minor system changes, such as correcting application software errors or minor processing adjustments, are normally handled as maintenance changes. Maintenance changes can require extensive technical knowledge, and some technical knowledge might be needed to classify a proposed change as major or minor. Will new processing requirements be the *straw that breaks the camel's back* in terms of hardware, network, and software capacity? Do proposed changes require application development tools that aren't compatible

with the current system's design or configuration? The answers to these questions determine whether the existing system will be modified or replaced by a new system.

If the existing system is to be modified, the application software components and files to be changed are identified, modified, tested, and deployed. The technical knowledge requirements depend heavily on the specific hardware, network, and software components affected by the change. If a new system is required, a new systems development life cycle is initiated.

Managing Computer Resources

So far, the need for technological knowledge has been discussed in the context of developing a single information system. However, think about the complexities and knowledge needed to manage the thousands of computer resources in a large organization, where many new development projects or system upgrades can be in progress at once.

In this type of environment, you must pay more attention to two critical technological issues—compatibility and future trends. Both issues are important because of the integration of computing technology in and across every function of modern organizations. For example, accounts payable and accounts receivable programs usually share a common hardware platform and operating system. Data from both systems is shared by a financial reporting system, which might be a different software system running on an entirely different computer. Data from many sources in the organization is often stored in a common database and accessed via an internal network or the Internet.

Managers of integrated collections of information systems and supporting infrastructure must contend with a great deal of technical complexity. They must ensure that each new system operates not only correctly by itself but also smoothly with all the other systems in the organization. They must also make sure hardware and software acquisitions are a good foundation for both current and future systems.

Given the rapid pace of change in computer technology, a manager must have a broad understanding of current technology and future technology trends. Will the computer purchased today be compatible with the hardware available three years from now? Can the organization's communication network be expanded easily to meet future needs? Should the organization invest only in tried-and-true technologies, or should it acquire cutting-edge technologies in hopes of improving performance or gaining a competitive advantage? Should the organization invest in enterprise-level software packages to use on local hardware, or should applications be deployed with a cloud service provider?

The answers to these questions require in-depth technical knowledge—far more knowledge than any one person has. Typically, managers confronted by these questions rely on the advice of experts and other sources of information. Even so, they must have an adequate base of technical knowledge to understand this information and advice.

SYSTEMS ARCHITECTURE AND COMPUTER TECHNOLOGY INFORMATION SOURCES

This book gives you a foundation of technical knowledge for a career in information system development or management. Unfortunately, this foundation will erode quickly because computer and information technologies change rapidly. How will you keep up with the changes?