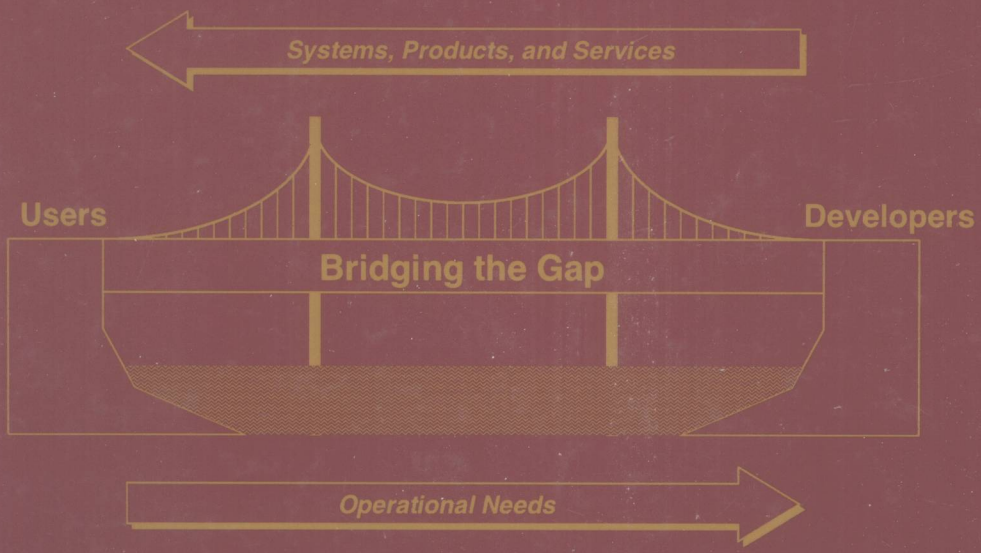
WILEY

System Analysis, Design, and Development

CONCEPTS, PRINCIPLES, AND PRACTICES



CHARLES S. WASSON

System Analysis, Design, and Development
Concepts, Principles, and Practices

Charles S. Wasson

WILEY-INTERSCIENCE

A John Wiley & Sons, Inc., Publication

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada.

Library of Congress Cataloging-in-Publication Data:
Wasson, Charles S., 1948–
System analysis, design, and development : concepts, principles, and practices / by Charles S. Wasson.
p. cm.
"A Wiley-Interscience publication."
Includes bibliographical references and index.
ISBN-13 978-0-471-39333-7
ISBN-10 0-471-39333-9 (cloth : alk. paper)
1. System design. 2. System analysis. I. Title.

QA76.9.S88W373 2005
004.2′1—dc22

2004061247

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

System Analysis, Design, and Development

Preface

As a user, acquirer, or developer of a system, product, or service, have you ever been confronted with one of the situations listed below?

- Wondered if the people who designed a product bothered to ask potential users to simply try it before selling it to the public.
- Found that during a major program review prior to component development that someone thought a requirement was so obvious it didn't have to be written down.
- Participated in a new system development effort and discovered at Contract Award that team members were already designing circuits, coding software, and developing mechanical drawings BEFORE anyone understood WHAT system users expected the system to provide or perform?
- Procured one of those publicized "designed for assembly" products and discovered that it was not designed for maintainability?
- Interacted with a business that employed basic business tools such as desktop computers, phones, and fax machines that satisfied needs. Then, someone decided to install one of those new, interactive Web sites only to have customers and users challenged by a "new and improved" system that was too cumbersome to use, and whose performance proved to be inferior to that of the previous system?

Welcome to the domain of system analysis, design, and development or, in the case of the scenarios above, the potential effects of the lack of System Engineering (SE).

Everyday people acquire and use an array of systems, products, and services on the pretense of improving the quality of their lives; of allowing them to become more productive, effective, efficient, and profitable; or of depending on them as tools for survival. The consumer marketplace depends on organizations, and organizations depend on employees to ensure that the products they produce will:

1. Perform planned missions *efficiently* and *effectively* when called upon.
2. Leverage user skills and capabilities to accomplish tasks ranging from simple to highly complex.
3. Operate using commonly available resources.
4. Operate *safely* and *economically* in their intended environment with *minimal* risk and intrusion to the general public, property, and the environment.
5. Enable the user to complete missions and return safely.
6. Be maintained and stored until the next use for low cost.
7. Avoid any environmental, safety, and health risks to the user, the public, or the environment.

In a book entitled *Moments of Truth*, Jan Carlzon, president of an international airline, observed that every *interaction* between a customer and a business through product usage or service support is a *moment of truth*. Each customer–product/service interaction, though sometimes brief, produces and influences perceptions in the User's mind about the system, products, and services of each

organization. Moment of truth interactions yield *positive* or *negative* experiences. Thus, the experiences posed by the questions above are *moments of truth* for the organizations, analysts, and engineers who develop systems.

Engineers graduate from college every year, enter the workforce, and learn system analysis, design, and development methods from the bottom up over a period of 10 to 30 years. Many spend entire careers with only limited exposure to the Users of their designs or products. As engineers are assigned increasing organizational and contract responsibilities, interactions with organizational customers also increase. Additionally, they find themselves confronted with learning *how to* integrate the efforts of other engineering disciplines beyond their field. In effect, they informally learn the rudiments of System Engineering, beginning with buzzwords, from the bottom up through observation and experience.

A story is told about an engineering manager with over 30 years of experience. The manager openly bragged about being able to bring in new college graduates, throw them into the work environment, and watch them sink or swim on their own without any assistance. Here was an individual with a wealth of knowledge and experience who was determined to let others "also spend 30 years" getting to comparable skill levels. Granted, some of this approach is fundamental to the learning experience and has to evolve naturally through personal *trials* and errors. However, does society and the engineering profession benefit from this type of philosophy.

Engineers enter the workplace from college at the lowest echelons of organizations mainly to apply their knowledge and skills in solving unique *boundary condition* problems. For many, the college dream of designing electronic circuits, software, or impressive mechanical structures is given a reality check by their new employers. Much to their chagrin, they discover that physical design is not the first step in engineering. They may be even startled to learn that their task is not to design but to find low-cost, acceptable risk solutions. These solutions come from research of the marketplace for existing products that can be easily and cost-effectively adapted to fulfill system requirements.

As these same engineers adapt to their work environment, they *implicitly* gain experience in the processes and methods required to transform a user's operational needs into a physical system, product, or service to fulfill contract or marketplace needs. Note the emphasis on *implicitly*. For many, the skills required to understand these new tasks and roles with increasing complexity and responsibility require tempering over years of experience. *If* they are fortunate, they may be employed by an organization that takes system engineering seriously and provides formal training.

After 10 years or so of experience, the demands of organizational and contract performance require engineers to assimilate and synthesize a wealth of knowledge and experience to formulate ideas about how systems operate. A key element of these demands is to communicate with their customers. Communications require open elicitation and investigative questioning, observation, and listening skills to understand the customer's operational needs and frustrations of unreliable, poorly designed systems or products that:

1. Limit their organization's ability to successfully conduct its missions.
2. Fail to start when initiated.
3. Fail during the mission, or cause harm to its operators, the general public, personal property, or the environment.

Users express their visions through operational needs for new types of systems that require application of newer, higher performance, and more reliable technologies, and present the engineer with the opportunity to innovate and create—as was the engineer's initial vision upon graduation.

Task leads and managers have a leadership obligation to equip personnel with the required processes, methods, and tools to achieve contract performance—for example, on time and within

budget deliverables—and enterprise survival over the long term. They must be visionary and proactive. This means providing just-in-time (JIT) training and opportunities to these engineers when they need these skills. Instead, they defer training to technical programs on the premise that this is on-the-job (OJT) training. Every program is unique and only provides a subset of the skills that SEs need. That approach can take years!

While browsing in a bookstore, I noticed a book entitled *If I Knew Then What I Know Now* by Richard Elder. Mr. Elder's book title immediately caught my attention and appropriately captures the theme of this text.

You cannot train experience. However, you can educate and train system analysts and engineers in system analysis, design, and development. In turn, this knowledge enables them to *bridge the gap* between a user's abstract operational needs and the hardware and software developers who design systems, products, and services to meet those needs. You can do this in a manner that avoids the *quantum leaps* by local heroes that often result in systems, products, or services that culminate in poor contract program performance and products that fail to satisfy user needs.

Anecdotal evidence suggests that organizations waste vast amounts of resources by failing to educate and train engineers in the concepts, principles, processes, and practices that consume on average 80% of their workday. Based on the author's own experiences and those of many others, if new engineers entering and SEs already in the workplace were equipped with the knowledge contained herein, there would be a *remarkable* difference in:

1. System development performance
2. Organizational performance
3. Level of personal frustrations in coping with complex tasks

Imagine the collective and synergistic power of these innovative and creative minds if they could be introduced to these methods and techniques without having to make quantum leaps. Instead of learning SE methods through informal, observational osmosis, and trial and error over 30+ years, *What if* we could teach system, product, or service problem-solving/solution development as an educational experience through engineering courses or personal study?

Based on the author's experience of over 30 years working across multiple business domains, this text provides a foundation in system analysis, design, and development. It evolved from a need to fill a void in the core curriculum of engineering education and the discipline we refer to as system engineering.

Academically, some people refer to System Engineering as an *emerging* discipline. From the perspective of specific engineering disciplines, System Engineering may be emerging only in the sense that organizations are recognizing its importance, even to their own disciplines. The reality is, however, the practice of engineering systems has existed since humans first employed tools to leverage their physical capabilities. Since World War II the formal term "system engineering" has been applied to problem solving-solution development methods and techniques that many specific engineering disciplines employ. Thus, system engineering concepts, principles, and practices manifest themselves in every engineering discipline; typically without the formal label.

In the chapters ahead, I share some of the *If I Knew Then What I Knew Now* knowledge and experiences. Throughout my career I have had the good fortune and opportunities to work and learn from some of the world's best engineering application and scientific professionals. They are the professionals who advanced the twentieth century in roles such as enabling space travel to the Moon and Mars, creating new building products and approaches, developing highly complex systems, and instituting high-performance organizations and teams.

This is a practitioner's textbook. It is written for advancing the *state of the practice* in the discipline we refer to as System Engineering. My intent is to go beyond the philosophical buzzwords

that many use but few understand and address the HOWS and WHYS of *system* analysis, design, and development. It is my hope that each reader will benefit from my discussions and will endeavor to expand and advance System Engineering through the application of the *concepts*, *principles*, and *practices* stated herein. Treat them as *reference guides* by which you can formulate your own approaches *derived* from and *tempered* by your own unique experiences.

Remember, every engineering situation is unique. As an engineer, you and your organization bear *sole responsibility and accountability* for the *actions* and *decisions* manifested in the systems, products, and services you design, develop, and deliver. Each user experience with those products and services will be a *moment of truth* for your organization as well as your own professional *reputation*. With every task, product, or service delivery, internally or externally, make sure the user's *moment of truth* is *positive* and *gratifying*.

ACKNOWLEDGMENTS

This work was made possible by the various contributions of the many people identified below.

My special debt of gratitude goes to Dr. Charles Cockrell, mentor, teacher, and leader; Neill B. Radke; Gerald "Jerry" Mettler; and Robert "Bob" M. Love who persevered through countless hours and iterations reviewing various sections of this work. Likewise, a special appreciation to Dr. Gregory M. Radecky for his technical counsel and commentary. Special thanks go to Sandra Hendrickson for support in revising the manuscript, to Lauren and Emily, and to Sharon Savage-Stull, and to Jean for coordinating the distribution of draft copies to reviewers. I thank members of the JPL—Brian Muirhead, Howard Eisen, David Miller, Dr. Robert Shisko, and Mary Beth Murrill—for sharing their time and experiences. Additionally, I thank Larry Riddle of the University of California, San Diego, and David Weeks for graphics submittals. Thanks also to INCOSE President-Elect Paul Robitaille and to William E. Greenwood and JoAnne Zeigler for their observations. To those true leaders who provided insightful wisdom, knowledge mentoring, training, concepts, and opportunities along my career path, I give a special word of *recognition* and *appreciation*. These include Bobby L. Hartway, Chase B. Reed, William F. Baxter, Dan T. Reed, Spencer and Ila Wasson, Ed Vandiver, and Kenneth King.

Finally, no words can describe how much I appreciate the dedication and caring of my loving wife and children who endured through the countless hours, weekends, and holidays and provided support over many years as this work evolved from concept to maturity. I couldn't have done this without you.

CHARLES S. WASSON
July, 2005

Table of Contents

Chapter 1

Introduction

1.1 FRAMING THE NEED FOR SYSTEM ANALYSIS, DESIGN, AND DEVELOPMENT SKILLS

One of the most perplexing problems with small, medium, or large system development programs is simply being able to deliver a *system*, *product*, or *service* without latent defects on schedule, within budget, and make a profit.

In most competitive markets, changes in technology and other pressures force many organizations to *aggressively* cut realistic schedules to win contracts to sustain business operations. Many times these shortcuts violate best practices through their elimination under the premise of "selective tailoring" and economizing.

Most programs, even under near ideal conditions, are often challenged to translate User needs into *efficient* and *cost-effective* hardware and software solutions for deliverable systems, products, and services. Technical program leads, especially System Engineers (SEs), create a strategy to bridge the gap. They translate the User's *abstract* vision into a language of specifications, architectures, and designs to guide the hardware and software development activities as illustrated in Figure 1.1. When aggressive "tailoring" occurs, programs attempt to bridge the gap via a quantum leap strategy. The strategy ultimately defaults into a continuous *build–test–redesign* loop until resources such as cost and schedules are overrun and exhausted due to the extensive rework. Systems delivered by these approaches are often patched and are plagued with undiscovered latent defects.

Bridging the gap between User needs and development of systems, products, and services to satisfy those needs requires three types of technical activities: 1) system analysis, 2) system design, and 3) system development (i.e., implementation). Knowledge in these areas requires education, training, and experience. Most college graduates entering the workforce do not possess these skills; employers provide very limited, if any, training. Most knowledge in these areas varies significantly and primarily comes from personal study and experience over many years. Given this condition, programs have the potential to be staffed by personnel lacking system analysis, design, and development skills attempting to make a quantum leap from user needs to hardware and software implementation.

Technically there are solutions of dealing with this challenge. This text provides a flexible, structural framework for "bridging the gap" between Users and system developers. Throughout this text we will build on workflow to arrive at the steps and practices necessary to plan and implement system analysis, design, and development strategy without sacrificing best practices objectives.

Part II *System Design and Development Practices* presents a *framework* of practice-based strategies and activities for developing systems, products, and services. However, system development requires more than simply implementing a standard framework. You must understand the

System Analysis, Design, and Development, by Charles S. Wasson

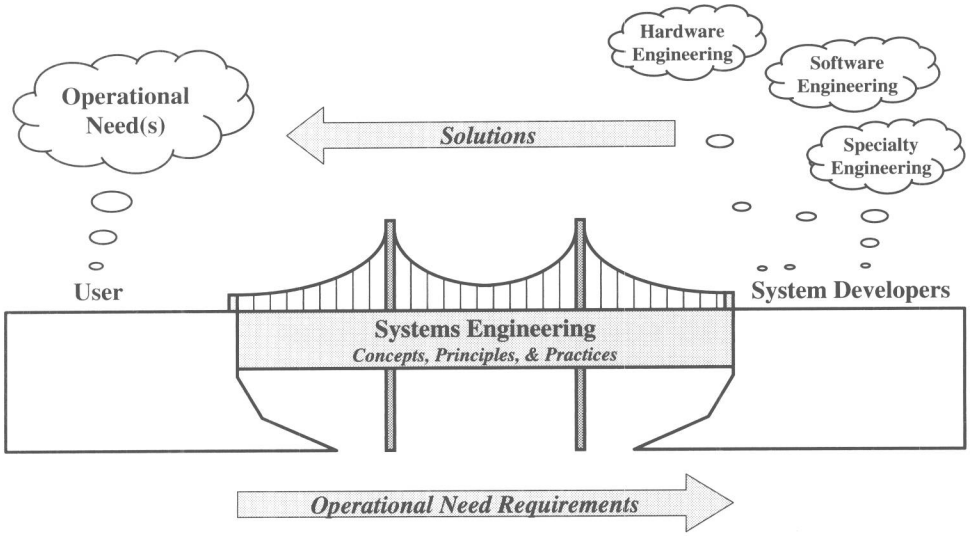


Figure 1.1 Systems Engineering—Bridging the Gap from User Needs to System Developers

foundation for the framework—HOW TO analyze systems. This requires understanding WHAT systems are; HOW the User envisions *deploying*, *operating*, *supporting*, and *disposing* of the system; under WHAT conditions and WHAT outcome(s) they are expected to achieve. Therefore, Part I addresses System Analysis Concepts as a precursor to Part 2.

This text identifies fundamental *system analysis*, *design*, and *development* practices that in the author's experiences are applicable to most organizations. The *concepts*, *principles*, and *practices* presented in Parts I and II represent a collection on topics that *condense* the *fundamentals* of key practices. Some of these topics have entire textbooks dedicated to the subject matter.

Your experiences may be different; that's okay. You and your organization are responsible and accountable for identifying the key concepts, principles, and practices unique to your line of business and programs and incorporate them into its command media—namely *policies* and *procedures*. Using this knowledge and framework, personnel at all levels of the organization are better postured to make informed decisions to bridging the gap from User needs to system, product, and service solutions to meet those needs without having to take a quantum leap.

Chapter 2

Book Organization and Conventions

2.1 HOW THIS BOOK IS ORGANIZED

There is a wealth of engineering knowledge that is well documented in textbooks targeted specifically for disciplinary and specialty engineers. In effect, these textbooks are compartmentalized bodies of knowledge unique to the discipline. The challenge is that SE requires knowledge, application, and integration of the concepts in these bodies of knowledge. The author's purpose in writing this book is not to duplicate what already exists but rather to complement and link SE and development to these bodies of knowledge as illustrated in Figure 2.1.

To accomplish these interdisciplinary linkages, the topical framework of the book is organized the way SEs think. SEs analyze, design, and develop systems. As such, the text consists of two parts: Part I *System Analysis Concepts* and Part II *System Design and Development Practices*. Each part is organized into series of chapters that address concepts or practices and include *Definitions of Key Terms* and *Guiding Principles*.

Part I: System Analysis Concepts

Part I provides the fundamentals in systems analysis and consists of a several series of topics:

- System entity concepts
- System architecture concepts
- System mission concepts
- System operations concepts
- System capability concepts

Each series within a part consists of chapters representing a specific topical discussion. Each chapter is sequentially numbered to facilitate quick location of referrals and topical discussions. The intent is to isolate topical discussions in a single location rather than a fragmented approach used in most textbooks. Due to the interdependency among topics, some overlap is unavoidable. In general, Part I provides the underlying foundation and framework of concepts that support Part II.

Unlike many textbooks, you will not find any equations, software code, or other technical exhibits in Part I. SE is a problem solving–solution development discipline that requires a fundamental understanding in HOW to think about and analyze systems—HOW systems are organized, structured, defined, bounded, and employed by the User.

System Analysis, Design, and Development, by Charles S. Wasson

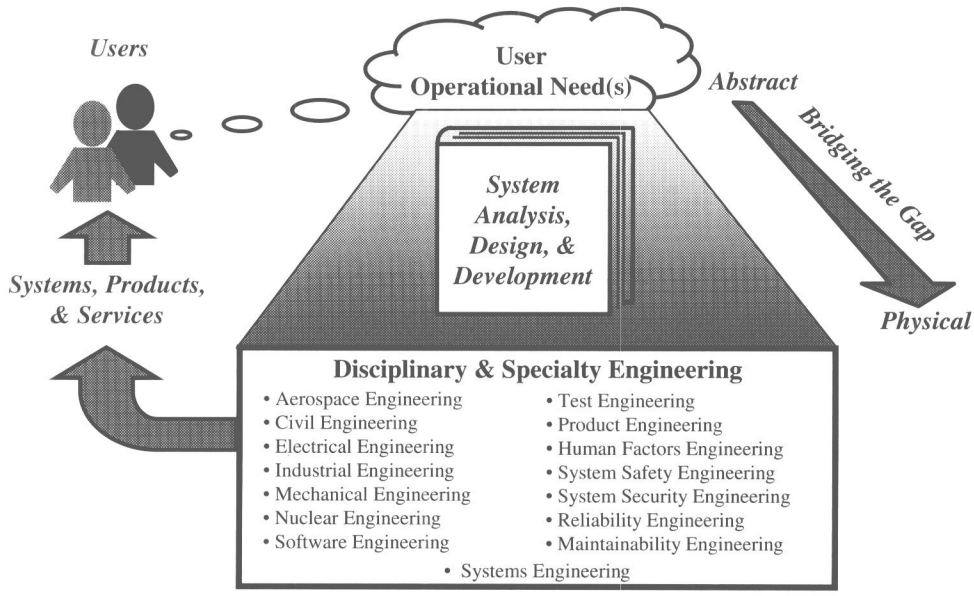


Figure 2.1 Book Scope

Part II: System Design and Development Practices

Part II builds on the *system analysis concepts* of Part I and describes the *system design and development practices* embodied by the discipline we refer to as system engineering. Part II contents consists of several *series* of practices that include:

- System development strategies
- System specification
- System design
- Decision support
- System verification and validation
- System deployment, operations, and support

Each series covers a range of topical practices required to support the series.

2.2 DEFINITIONS

SE, as is the case with most disciplines, is based on *concepts*, *principles*, *processes*, and *practices*. The author's context for each of these terms can be better understood as follows:

- **Concept** A visionary expression of a proposed or planned action that leads to achievement of a disciplinary objective.
- **Principle** A guiding thought based on empirical deduction of *observed behavior* or *practices* that proves to be true under most conditions over time.
- **Process** A sequence of *serial* and/or *concurrent* operations or tasks that transform and/or add value to a set of inputs to produce a product. Processes are subject to external *controls* and *constraints* imposed by regulation and/or decision authority.

- **Operation** A collection of outcome-based tasks required to satisfy an operational objective.
- **Task** The application of methods, techniques, and tools to add value to a set of inputs—such as materials and information—to produce a work product that meets "fitness for use" standards established by formal or informal agreement.
- **Practice** A systematic approach that employs *methods* and *techniques* that have been demonstrated to provide results that are generally *predictable* and *repeatable* under various operating conditions. A practice employs processes, operations, or tools.
- **Best or Preferred Practice** A practice that has been *adopted* or *accepted* as the most suitable method for use by an organization or discipline. Some individuals rebuff the operative term "best" on the basis it is relative and has yet to be universally accepted as THE one and only practice that is above all others. Instead, they use *preferred* practice.

2.3 TEXT CONVENTIONS

This textbook consists of several types of annotations to facilitate readability. These include referrals, author's notes, guideposts, reference identifiers, and examples. To better understand the author's context of usage, let's briefly summarize each.

Referrals. SE concepts, processes, and practices are highly interdependent. Throughout the book you will find *Referrals* that suggest related chapters of the book that provide additional information on the topic.

Author's Notes. *Author's Notes* provide insights and observations based on the author's own unique experiences. Each *Author's Note* is indexed to the chapter and in sequence within the chapter.

Guideposts. *Guideposts* are provided in the text to provide the reader an understanding of WHERE you are and WHAT lies ahead in the discussion. Each *guidepost* is indexed to the chapter and sequence within the section.

Reference Identifiers. Some graphics-based discussions progress through a series of steps that require navigational aids to assist the reader, linking the text discussion to a graphic. *Reference Identifiers* such as (#) or circles with numbers are used. The navigational reference IDs are intended to facilitate classroom or training discussions and reading of detailed figures. It is easier to refer to "Item or ID 10" than to say "system development process."

Examples. *Examples* are included to illustrate *how* a particular concept, method, or practice is applied to the development of real world systems. One way SEs deal with *complexity* is through concepts such as *abstraction*, *decomposition*, and *simplification*. You do not need a Space Shuttle level of complexity example to learn a key point or concept. Therefore, the examples are intended to accomplish one objective—to *communicate*. They are not intended to insult your intelligence or impress academic egos.

References. Technical books often contain pages of references. You will find a limited number of references here. Where external references are applicable and reinforce a point, explicit call-outs are made. However, this is a *practitioner's* text intended to equip the reader with the practical knowledge required to perform system analysis, design, and development. As such, the book is

intended to stimulate the reader's thought processes by introducing fresh approaches and ideas for advancing the state of the practice in System Engineering as a professional discipline, not summarizing what other authors have already published.

Naming Conventions. Some discussions throughout the book employ terms that have *generic* and *reserved* word contexts. For example, terms such as equipment, personnel, hardware, software, and facilities have a generic context. Conversely, these same terms are considered SE system elements and are treated as RESERVED words. To delineate the context of usage, we will use lowercase spellings for the generic context and all capitals for the SE unique context—such as EQUIPMENT, PERSONNEL, HARDWARE, SOFTWARE, and FACILITIES. Additionally, certain words in sentences require communication emphasis. Therefore, some words are *italicized* or CAPITALIZED for emphasis by the author as a means to enhance the readability and communicate key points.

2.4 GRAPHICAL CONVENTIONS

System analysis and *design* are graphics-intensive activities. As a result a standard set of graphical conventions is used to provide a level of continuity across a multitude of highly interdependent topics. In general, system analysis and design employ the following types of relationships:

1. Bounding WHAT IS/IS NOT part of a system.
2. Abstractions of collections of entities/objects.
3. Logical associations or relationships between entities.
4. Iterations within an entity/object.
5. Hierarchical decomposition of abstract entities/objects and integration or entities/objects that characterized by *one-to-many* and *many-to-one* entity or object relationships—for example, parent or sibling.
6. Peer-to-peer entity/object relationships.
7. Time-based, *serial* and *concurrent* sequences of *workflow*, and *interactions* between entities.
8. Identification tags assigned to an entity/object that give it a unique identity.

There are numerous graphical methods for illustrating these relationships. The Object Management Group's (OMG) *Unified Modeling Language* (UML®) provides a diverse set of graphical symbols that enable us to express many such relationships. Therefore, diagrams employing UML symbology are used in this book WHERE they enable us to better communicate key concepts. UML annotates *one-to-many* (i.e., *multiplicity*) entity relationships with "0 . . . 1," "1," "1 . . .*," and so forth. Many of the graphics contain a significant amount of information and allow us to forgo the multiplicity annotations. Remember, this text is intended to communicate concepts about system analysis, design, and development; not to make you an expert in UML. Therefore, you are encouraged to visit the UML Web site at www.omg.org for implementation specifics of the language. Currently, SE versions of UML®, SYSML, is in the process of development.

System Block Diagram (SBD) Symbology

One of the first tasks of system analysts and SE is to bound WHAT IS/IS NOT part of the system. System Block Diagrams (SBDs), by virtue of their box structure, offer a convenient way to express these relationships, as illustrated at the left side of Figure 2.2.