Chapter 1

Barry W. Boehm and Kevin J. Sullivan. *Software Economics.* White Paper, 1999. Boehm and Sullivan stress the importance of understanding the relationships between software development decisions and business decisions. The goal is maximal value creation. Technical software decisions should be based on value creation; therefore, software engineers, designers, and managers should base decisions on value maximization objectives. Boehm and Sullivan assert that software engineers are usually not involved in and do not understand **enterprise**-level value creation objectives, and that senior management often does not understand success criteria for software development or how investments at the technical level can contribute to value creation. It is in our enlightened self-interest to increase our understanding of and ability to deal with the economic aspects of software and its development.

Dennis M. Buede. *The Engineering Design of Systems: Models and Methods.* New York: John Wiley & Sons, 2000. This book provides an excellent discussion of systems engineering, with good emphasis on requirements, the design process, modeling, architecture development, and decision analysis. Buede emphasizes that requirements are the cornerstone of the systems engineering process. Case studies and problems are provided.

W. Edwards Deming. *Out of the Crisis.* Cambridge, MA: Massachusetts Institute of Technology, Center for Advanced Engineering Study, 1986. Dr. Deming is the father of quality in Japan and did much for the United States as we reluctantly gave more attention to it. Deming's 14 points provide a theory of management. His seven deadly diseases afflict most companies and stand in the way of progress. Deming's thesis is that American management does not enable and empower its work force, which is "only doing its best." Deming asserts that American management does not understand variation (faults of the system [common causes] and faults from fleeting **events** [special causes]). It's vital for every manager to capture the essence of Deming's perspective (see the reference to Mary Walton's book later in this section).

Donald C. Gause and Gerald M. Weinberg. *Exploring Requirements: Quality Before Design.* New York: Dorset House Publishing, 1989. This book is one of the classics concerning requirements elicitation. Gause and Weinberg provide a large collection of ideas and approaches based on their consulting experience in discovering what is desired in systems. They believe that meeting the following three conditions helps to ensure a successful project: (1) achieving a consistent

understanding of the requirements among all participants, (2) achieving teamwork, and (3) knowing how to work effectively as a team (that is, developing the necessary skill and tools to define requirements).

Tom Gilb. *Principles of Software Engineering Management.* Wokingham, UK: Addison-Wesley, 1988. Gilb has made many significant contributions to the industry over the years, including his work concerning inspections, requirements-driven management, evolutionary delivery, and impact estimation. His book is an easily readable, comprehensive discussion of software engineering principles and is recommended as a good foundation for practitioner study. See Gilb's Web site for current activities and publications (http://www.Result-Planning.com).

Jeffrey O. Grady. *System Requirements Analysis.* New York: McGraw-Hill, 1993. Grady has devoted his career to requirements and related topics, and he teaches courses and develops tutorials. He is a founding member of the National Council on Systems Engineering (NCOSE; now the International NCOSE). This book provides a systems approach and thorough explanations of **requirements analysis,** traceability specifications, requirements integration, **requirements verification,** and explanations of techniques including structured analysis, structured **decomposition,** and architecture synthesis.

Ivy F. Hooks and Kristin A. Farry. *Customer-Centered Products: Creating Successful Products Through Smart Requirements Management.* New York: AMACOM, 2001. This book is a guide to how good requirements are possible when managers are involved in guiding the requirements process. It provides advice and insights based on years of experience and many suggestions for how to perform requirements-related activities. Hooks and Farry emphasize the importance of managers allocating resources, defining and enforcing processes, educating personnel, and measuring the impact of requirements on final product quality. This is recommended reading for project managers.

Capers Jones. "Software Project Management in the 21st Century." *American Programmer* 1998:11(2):24–30. Jones notes that manual software estimating methods fail for large systems because of their **complexity**. He advocates using automated project management and software cost-estimating tools, noting that integrated management tool suites that share common **interfaces** and common data repositories will probably be developed within a few years. He suggests that a Web-based software cost estimation and planning service and a benchmarking service are likely. He indicates that the most visible and important gap in software

project management capabilities concerns data and information and that there is a growing need to be able to deal with long-range estimating and measurement at the enterprise or corporate level.

Capers Jones. *What It Means To Be "Best in Class" for Software.* Burlington, MA: Software Productivity Research, Inc. Vers. 5. February 10, 1998. This rigorous and analytical report lists 15 key software performance goals that, if achieved, indicate that *best-in-class* status is within your grasp. The report provides qualitative and quantitative results from the top 5% of the projects of the top 10% of the clients of Software Productivity Research, Inc. (SPR). Also, quality targets for the five levels of the Software Engineering Institute CMM are suggested. The report surveys process improvement strategies and tactics that excellent software producers have utilized. The report is available from SPR at capers@spr.com.

Dean Leffingwell and Don Widrig. *Managing Software Requirements: A Unified Approach.* Reading, MA: Addison-Wesley, 2000. This book emphasizes the team skills that are required for the requirements process. Leffingwell and Widrig share their many years of experience and describe proven techniques for understanding needs, organizing requirements information, and managing the scope of a project.

James N. Martin. *Systems Engineering Guidebook: A Process for Developing Systems and Products.* Boca Raton, FL: CRC Press, 1996. This book is a comprehensive guide to the systems engineering process, application tasks, methodologies, tools, documentation, terminology, system hierarchy, synthesis of functional and performance requirements into the product architecture, integration, verification, metrics, **defects** and defect types, and programmatic application. Martin describes the systems engineering process as a multidisciplinary effort. The system life cycle and **life cycle model** are described, and the role of the systems engineering process champion is explained. An extensive set of figures and tables that describe all aspects of systems engineering is provided.

Mark C. Paulk, Bill Curtis, Mary Beth Chrissis, and Charles V. Weber. *Capability Maturity Model for Software.* Version 1.1. Pittsburgh, PA: Software Engineering Institute, Carnegie-Mellon University, 1993. Also available at http://www.sei.cmu.edu/publications/documents/93.reports/93.tr.024.html. The CMM for Software (SW-CMM) has been the industry model for software process improvement for more than a decade. It has provided a standard that has allowed projects and organizations to evaluate their practices, provide a basis for improvement actions, develop improvement plans, and measure improvements. This book is highly recommended for projects and organizations.

Ian Sommerville and Pete Sawyer. *Requirements Engineering: A Good Practice Guide.* New York: John Wiley & Sons, 1997. This book provides a set of basic, intermediate, and advanced guidelines consistent with SW-CMM levels to facilitate gaining requirements engineering process maturity, and it explains in detail how to apply these guidelines in practice. Sommerville and Sawyer define requirements elicitation as follows: The system requirements are *discovered* (emphasis added) through consultation with stakeholders, from system documents, domain knowledge, and market studies. Their Chapter 4 explains how to apply 13 guidelines to perform requirements elicitation.

Mary Walton. *The Deming Management Method.* New York: Putnam Publishing, 1986. This is a good summary and explanation of Dr. Deming's teachings, with a foreword by the master. Walton describes Deming's 14 points, the seven deadly diseases, and the parable of the red beads. Walton describes Deming's beliefs that management is the primary cause of the results in organizations. The "workers" (everyone else) are powerless, lacking the environment to be effective. Managers should be challenged to recognize the distinction between a stable system and an unstable one and to be able to recognize and address special causes. The conditions necessary to achieve teamwork are described. This is recommended reading for anyone who seeks a good foundation for a quality improvement ethic.

Bill Wiley. *Essential System Requirements: A Practical Guide to Event-Driven Methods.* Reading, MA: Addison-Wesley, 2000. Wiley provides an event-driven strategy for software development that he believes provides an intuitive, effective partitioning of the user domain and the proposed system. Events jump-start the identification and specification of system requirements with early user involvement and improved user communication. When combined with a spiral, incremental approach that dovetails with an "architected" rapid application development strategy, event-driven methods accelerate the delivery process. Wiley provides a **function point** example.

Chapter 2

Frank Carr, Kim Hurtado, Charles Lancaster, Charles Markert, and Paul Tucker. *Partnering in Construction: A Practical Guide to Project Success.* Chicago, IL: American Bar Association Publishing, 1999. This book is an excellent guide for implementing the partnering process. Carr and colleagues are pro-

fessional facilitators with extensive experience in actual partnering efforts. They provide a practical approach that includes samples of the products of partnering workshops. Lessons learned and case studies from their experience are provided. A glossary of related terms is included.

Michael Cusumano and Richard Selby. *Microsoft Secrets: How the World's Most Powerful Software Company Creates Technology, Shapes Markets, and Manages People.* New York: Free Press, 1995. Cusumano and Selby performed almost two years of on-site research at Microsoft headquarters. The book is an objective, analytical, and thorough profile of an important company. It focuses on the relationship between business strategies and software development. It explains Microsoft's management model, organizational culture, technologies, and software development approach.

Richard Harwell. "System Engineering Is More Than Just a Process." In: Martin, James N., ed. *Systems Engineering Guidebook.* Boca Raton, FL: CRC Press, 1996: 249–255. Harwell's work advocates several of the effective requirements practices recommended in this book, noting that many companies overlook them. As a result, those companies realize no apparent benefit from the system engineering process. Harwell emphasizes that today's customer and the development team must work together to achieve a successful development process.

Capers Jones. *Assessment and Control of Software Risks.* Englewood Cliffs, NJ: Prentice Hall, 1994. Jones cataloged 63 specific risks that affect software projects. He provides a description of each risk, severity, frequency, occurrence, root causes, cost impact, methods of prevention, methods of control, product support, and the effectiveness of known therapies. Jones discusses the risk of creeping requirements, inaccurate sizing of deliverables, and crowded office conditions, to name a few examples. His book is a valuable reference. As Jones maintains, problems don't go away by themselves, and his comprehensive treatment provides valuable insights. He provides a list of tools that he believes is an approximate 5% sample of thousands of commercially available tools.

Charles Markert. *Partnering: Unleashing the Power of Teamwork.* 1998. Available at markert@erols.com. This marketing briefing describes partnering; its attributes, characteristics, and benefits; and why one would participate in partnering. It also provides a process for partnering: preparation, a partnering workshop, what it means to "walk the talk," getting feedback, and celebration. Keys to

the partnering process are presented, and lessons learned from partnering experiences are presented.

Gerald M. Weinberg, James Bach, and Naomi Karten, eds. *Amplifying Your Effectiveness.* New York: Dorset House, 2000. The editors and a group of software consultants present ideas on how software engineers and managers can amplify their professional effectiveness—as individuals, as members of teams, and as members of organizations. The contributed essays are organized in the categories of empowering the individual, improving interpersonal interactions, mastering projects, and changing the organization. A theme is that we're more likely to enhance effectiveness if we start by looking within and asking ourselves what we might do better or differently. There are a lot of insights here, and they are presented in a fresh, unique way.

Karl E. Wiegers. *Software Requirements.* Redmond, WA: Microsoft Press, 1999. This is an excellent, easily readable book that offers practical advice on how to manage and participate in the requirements engineering process. Commitment is viewed in this book as "finding the voice of the customer." Engaging the participants in the project is considered critical to success. Wiegers's experience at Eastman Kodak and as an independent consultant indicates that customers don't know what they really need, and neither do the developers.[7]

Chapter 3

Warren Bennis and Patricia Ward Biederman. *Genius: The Secrets of Creative Collaboration.* Reading, MA: Perseus Books, 1997. This is a book about great teams. I found the "take-home lessons" at the end of the book to be apt descriptions of the most successful teams with which I have been associated. One of the great teams described in this book is The Skunk Works, a term that has become synonymous with secret, groundbreaking technological work. Another is The Manhattan Project, the story of the building of the atomic bomb. "None of us is as smart as all of us" is a valuable lesson.

Tom DeMarco and Timothy Lister. *Peopleware: Productive Projects and Teams.* 2nd ed. New York: Dorset House Publishing, 1999. This book is a classic that presents the human dimension concerning technical development projects. DeMarco and Lister have collected data since 1977 concerning development projects and their results and have more than 500 project histories in their data-

base. A premise is that the large number of failures of projects of all sizes is not the result of failure of technology, but rather human issues such as project management, communications, high turnover, and lack of motivation contribute to project failure. This second edition provides the text of the first and adds eight new chapters. Readers will discern a host of insights that will contribute to more effective teams and project management. Management's challenge is to create a culture that allows people to work effectively. This book is an important read for any project manager.

Roger Fisher and William Ury. *Getting to Yes.* New York: Penguin Books, 1991. Reaching agreement on a specification is a process of negotiating the interests of several parties. This book emphasizes that reaching agreement is more important than risking inflexibility. Negotiation skills to achieve win-win agreements speed up the process and benefit everyone. Fisher and Ury emphasize that it's more important to understand the other party's underlying interests than to focus on a specific position.

Luke Hohmann. *Journey of the Software Professional: A Sociology of Software Development.* Upper Saddle River, NJ: Prentice Hall PTR, 1997. This is a book of practical advice for developers and managers who are serious about enhancing their own effectiveness and the effectiveness of their teams. It addresses many of the human or "soft" issues that are so critical in building systems successfully. It provides suggestions for career development, training, development teams, interpersonal relations, communications, and organizational structures, among others.

Watts S. Humphrey. *Introduction to the Team Software Process.* Reading, MA: Addison-Wesley, 2000. Humphrey is a superb writer, and this book shares his tremendous insights into what makes software teams work effectively. Although it builds on the personal software process, it's a valuable resource for anyone seeking to improve team effectiveness. Humphrey provides some practical strategies for facilitating teamwork, such as

- Eliminating work that gets in the way of improving the product
- Effectively applying work habits and processes
- Systematically tracking project progress
- Creating an environment that fosters unbridled enthusiasm
- Using *milestone scheduling* to keep schedules aggressive but not unrealistic
- Aligning personal growth goals with the necessary work
- Promoting beneficial attitudes such as a continuous improvement ethic

Humphrey's final chapter, Teamwork, is insightful. This book is a valuable read for any project manager.

Dean Leffingwell. *Calculating Your Return on Investment from More Effective Requirements Management.* (Available at http://www.rational.com/products/whitepapers/300.jsp.) This reference provides an easy-to-use way to calculate savings from requirements-related efforts. It may be helpful to you if your manager requests data concerning the value of efforts involving the requirements process.

Mark C. Paulk. The "Soft Side" of Software Process Improvement. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1999. Paulk is the leading author of the Capability Maturity Model for Software (SW-CMM) and has deep sensitivity to the people-related aspects of the industry. Among many valuable insights addressed in his briefing are (1) the best people outperform the worst by approximately 10:1; (2) the best performer is 2.5 times more productive than the median performer; (3) CMM levels 4 and 5 organizations tend to have required training in team building, negotiation skills, interpersonal skills, domain knowledge, management skills, and technical skills; (4) people tend to be overly optimistic about what they can do; and (5) managers tend to ignore possible events that are very unlikely or very remote, regardless of their consequenses. Paulk provides several charts concerning the People CMM. Perhaps most germane to this chapter is that the range in the performance of teams can be 200:1 and that united teams are extraordinarily powerful.

Roger S. Pressman. *Software Engineering: A Practitioner's Approach.* 4th ed. New York: McGraw Hill, 1997. Pressman is an extremely knowledgeable, internationally known consultant and author who has extensive practical experience as an industry practitioner and manager. He specializes in helping organizations establish effective practices, and he developed an assessment method that helps clients assess their current state of engineering practice. Pressman's Web site provides extensive resources, including 30 templates for engineering documents. See http://www.rspa.com.

Chapter 4

Barry Boehm. WinWin Spiral Model & Groupware Support System. 1998 Available at http://sunset.usc.edu/research/WINWIN/index.html. Dr. Boehm is Director of the University of Southern California Center for Software Engi-

neering. The Center is under contract to the Defense Advanced Research Projects Agency via the Air Force Research Laboratory (formerly known as Rome Laboratories). It plans to develop (in collaboration with The Aerospace Corporation) a robust version of the WinWin System and to apply it to the domain of satellite ground stations.

Barry Boehm, Alexander Egyed, Julie Kwan, Dan Port, Archita Shah, and Ray Madachy. "Using the WinWin Spiral Model: A Case Study." ***IEEE Computer*** **1998:31 33–44.** This University of Southern California Center for Software Engineering research project has three primary elements: (1) Theory W, a management theory and approach that says making winners of the system's key stakeholders is a necessary and sufficient condition for project success; (2) the WinWin Spiral Model, which extends the spiral software development model by adding Theory W activities to the front of each cycle; and (3) WinWin, a groupware tool that makes it easier for distributed stakeholders to negotiate mutually satisfactory system specifications. The authors found in this work that the most important outcome of product definition is not a rigorous specification but a team of stakeholders with enough trust and shared vision to adapt effectively to unexpected changes. The researchers believe that the approach will transition well to industry use.

Daniel P. Freedman and Gerald M. Weinberg. ***Handbook of Walkthroughs, Inspections, and Technical Reviews.*** **3rd ed. Chicago: Scott, Foresman and Co., 1990.** This book provides a variety of examples of peer reviews and is a good source for organizations that want to consider alternative approaches.

Donald C. Gause and Gerald M. Weinberg. ***Are Your Lights On? How to Know What the Problem REALLY Is.*** **2nd ed. New York: Dorset House Publishing, 1989.** As the title suggests, this book is interesting and light reading but offers valuable insights concerning real needs. The authors' perspective is that customers need assistance in defining their real requirements. A good requirements process will (1) identify the real problem, (2) determine the problem's "owner," (3) identify its root cause, and (4) determine whether to solve it. This is recommended reading for requirements engineers and their customers.

Tom Gilb and Dorothy Graham. ***Software Inspection.*** **Reading, MA: Addison-Wesley, 1993.** This book is about inspections of any work product, not just software. The authors' approach is very rigorous and therefore requires more training and is more expensive than normal peer reviews. However, it results in more defects being removed earlier, thus saving costs later in the development

cycle. This book is invaluable for an organization that is committed to using inspections of work products—a proven method with good payback. Note that Rob Sabourin offers an economical inspections training and implementation approach. Contact him at rsabourin@amibug.com.

Rita Hadden. "How Scalable Are CMM Key Practices?" *CROSSTALK* 1998: vol. 11(4) 18–23. Hadden provides process improvement consulting services for organizations of all sizes. She notes that many practitioners are convinced that models such as the SW-CMM are not practical for small organizations because the cost of applying the recommended practices outweighs benefits. Her experience with more than 50 small projects does not support this view. The article describes using a disciplined, repeatable approach for projects of short duration. She concludes that CMM key practices are scalable.

Ivy Hooks. *Guide for Managing and Writing Requirements.* 1994. Available at ivyh@complianceautomation.com. This is a concise, well-written guidebook based on extensive experience by a practicing requirements engineer and consultant. It addresses scoping a project, managing requirements, how systems are organized, and levels of requirements, writing good requirements, requirements attributes, and specifications.

Ivy Hooks. *Managing Requirements.* 1994. This white paper is available at the Compliance Automation Web site http://www.complianceautomation.com/. It provides a good analysis of how failure to invest in the requirements process affects projects, and it describes major problems based on Hooks's experience. Also, it describes some of the characteristics of good requirements.

Ivy Hooks. *Writing Good Requirements: A One-Day Tutorial.* McLean, VA, 1997 Compliance Automation, Inc. Sponsored by the Washington metropolitan area chapter of the International Council on Systems Engineering, June 1997. This is an example of the types of briefings and courses that can be provided to facilitate a project or an organization in dealing with the requirements process. The pearl here is to ensure that you *have* a requirements process and that you take advantage of industry best practices in executing it. Don't find your own way and learn the errors of your ways at considerable financial, personal, project, and organizational costs.

Pradip Kar and Michelle Bailey. *Characteristics of Good Requirements.* 1996. Available at http://www.complianceautomation.com/. This document provides a

valuable, readily available discussion of the characteristics of individual and aggregate requirements (note that characteristics of individual requirements are applicable to aggregates too). Kar and Bailey emphasize that writing good requirements is difficult, requires careful thinking and analysis, but is not magical. Time spent up front, carefully defining and articulating the requirements, is essential to ensuring a high-quality product. This is recommended reading for requirements engineers.

Joachim Karlsson and Kevin Ryan. "A Cost-Value Approach for Prioritizing Requirements." ***IEEE Software*** **14(5) 1997: 67–74.** This is an excellent article that explains a method for prioritizing requirements (see the summary of their method provided in this chapter). Karlsson and Ryan provide a process for using the cost-value approach, utilizing the Analytic Hierarchy Process (AHP), which is also explained in their article.

Geri Schneider and Jason P. Winters. ***Applying Use Cases: A Practical Guide.*** **Reading, MA: Addison-Wesley, 1998.** This is a practical guide to developing and using use cases. Schneider and Winters provide examples from their experience and provide a case study that offers insight into common errors. An illustration of the UML notation for diagramming use cases is provided. Of particular use to requirements engineers is a "how-to" discussion on applying use cases to identify requirements.

I. Sommerville, P. Sawyer, and S. Viller. "Viewpoints for Requirements Elicitation: A Practical Approach." In: ***Proceedings of the 1998 International Conference on Requirements Engineering*** **(ICRE'98), April 6–10, 1998, Colorado Springs, CO. New York: IEEE Computer Society, 1998: 74–81.** Sommerville and colleagues introduce an approach called PREview to organize requirements derived from radically different sources. They show how concerns that are key business drivers of the requirements elicitation process may be used to elicit and validate system requirements. They note that PREview has been designed to allow incremental requirements elicitation (see Figure 4-12 for a high-level view of the PREview process).

Gerald M. Weinberg. ***The Secrets of Consulting.*** **New York: Dorset House Publishing, 1986.** Weinberg defines consulting as the art of influencing people at their request. As noted by Virginia Satir in the foreword, this book actually advises people on how they can take charge of their own growth. The author provides a light-hearted view of the role of a consultant, sharing valuable insights

about people. A fundamental tenet is that we all need to follow a personal learning program. Several sources for readings and other experiences are provided.

Karl Wiegers. "First Things First: Prioritizing Requirements." *Software Development Magazine* 1999: 7(10):24–30. This is a good explanation of why requirements need to be prioritized and a helpful description of how to do it. Wiegers provides a Microsoft Excel requirements prioritization spreadsheet and other requirements tools that can be downloaded from his Web site, http://www.processimpact.com.

Karl Wiegers. "Habits of Effective Analysts." *Software Development Magazine* 2000: 8(10):62–65. See also http://www.swd.mgazine.com. Wiegers provides thoughtful and provocative insights concerning the role of the requirements engineer (also called the requirements analyst, business analyst, systems analyst, or requirements manager), patterned after Steven Covey's acclaimed book *The Seven Habits of Highly Effective People* (Fireside, 1989). He emphasizes that requirements engineering has its own skill set and body of knowledge, which is given scant attention in most computer science educational curricula and even by most systems and software engineering organizations. Many organizations expect developers or project managers to handle this vital function on their own. A competent requirements engineer must combine communication, facilitation, and interpersonal skills with technical and business domain knowledge. Even a dynamite developer or a systems-savvy user needs suitable preparation before acting in this role. Wiegers recommends that every organization should develop an experienced cadre of requirements analysts, even though requirements engineering may not be a full-time function on every project. This article is recommended reading for all PMs and task leaders.

Chapter 5

The CMMI project is a collaborative effort sponsored by the U.S. DoD Office of the Secretary of Defense/Acquisition, Technology, and Logistics and the National Defense Industrial Association (NDIA), with participation by government, industry, and the SEI. The project's objective is to develop a product suite that provides industry and government with a set of integrated products to support process and product improvement. The intent is to preserve government and industry investment in PI and to enhance the use of multiple models. The project's outputs will be integrated models, assessment methods, and training mate-

rials. The DoD's concerns were to stop proliferation of CMMs and to standardize one model. Work continues at a frantic pace on this project. However, because industry has a lot of effort and money invested in the SW-CMM (and to a lesser extent, the SE-CMM), implementation may not proceed as quickly as some anticipate. See http://www.sei.cmu.edu/cmm/cmms/cmms.integration.html.

Peter DeGrace and Leslie Hulet Stahl. *Wicked Problems, Righteous Solutions.* Englewood Cliffs, NJ: Yourdon Press, 1990. The authors look at the assumptions and expectations associated with life cycle models such as waterfall, incremental, spiral, and "all at once." They analyze the pros and cons of prototyping. They enable one to look at "wicked" software problems with a different perspective. Their view is that we are lucky as developers to get 90% of the most important requirements at the outset of a project (p. 69).

EIA. *ANSI/EIA 632, Processes for Engineering a System.* Arlington, VA: EIA, 1998. EIA 632 came about because the U.S. DoD determined in 1994 that MIL-STD-499B would not be released as a military standard. EIA's Committee on Systems Engineering (the EIA G-47 Committee) agreed to undertake the task of "demilitarizing" 499B and releasing it as an industry standard. The intent was to revise the military version in accordance with commercial practices to broaden the suitability of the standard for other government agencies and commercial industry. EIA 632 provides a comprehensive, structured, disciplined approach for all life cycle phases. The systems engineering process is applied iteratively throughout the system life cycle. Key aspects of industry's initiatives are captured to identify and integrate requirements better and to implement multidisciplinary teamwork, including potential suppliers, early in establishing the requirements. Other key aspects include establishing clear measurements of system responsiveness, encouraging innovation in products and practices, and focusing on process control rather than inspection. Also, risk management is encouraged.

EIA. *EIA/IS 731, Systems Engineering Capability.* Arlington, VA: EIA, 1998. The EIA G-47 Committee initiated an effort to merge the INCOSE SECAM and the EPIC SE-CMM in 1996. EIA interim standard (IS) 731, Version 1.0, was released on January 20, 1998. It contained two parts: Part 1 was the Systems Engineering Capability Model (SECM), and Part 2 was the SECM Appraisal Method. The purpose of this standard is to support the development and improvement of system engineering. Attention to this standard has become overcome by events

because of the CMMI initiative, being driven by the U.S. DoD and being worked by the NDIA by representatives of EIA and the SEI.

Enterprise Process Improvement Collaboration (EPIC). *A Systems Engineering Capability Maturity Model.* Version 1.1. Pittsburgh, PA: SEI, Carnegie-Mellon University, 1995. To download a copy, visit http://www.sei.cmu.edu/publications/documents/95.reports/95.mm.003.html. Following the development of the SW-CMM, some systems engineers determined that they would create a similar model for systems engineering. Version 1.0 of the SE-CMM was piloted in 1994, and Version 1.1 was released on November 1, 1995. This model is extremely useful. It contains 18 PAs, each with base practices. An SE-CMM Appraisal Method (SAM) was also developed. SAM Version 1.1 is dated March 1996. It too was developed by the SE-CMM collaboration members, primarily systems engineers from major companies. See http://www.sei.cmu.edu/publications/documents/94.reports/94.hb.005.html.

J. Davidson Frame. *Managing Projects in Organizations.* Rev. ed. San Francisco, CA: Jossey-Bass Publishers, 1995. Frame provides a practical, hands-on approach with attention to behavioral aspects. He emphasizes the importance of ensuring that the project is based on a clear need and specifying what the project should accomplish. He addresses the importance of requirements in the context of real needs.

Litton PRC. *Phoenix Software Process Improvement Reference Guide.* 3rd ed. McLean, Virginia: Litton PRC, April 1996. This is a desk guide for developers that summarizes the corporate PI program. Such a book facilitates PI by providing a readily available source of policies, processes, resources available, schedule of PI activities, training courses, metrics, acronyms, work breakdown structure for the PI program, and examples of formats to be used (for example, for a project PI plan). It begins with a discussion of senior management sponsorship and oversight of the PI efforts and addresses many mechanisms that have been put in place to maintain the momentum of the PI program. It's recommended that organizations consider providing a similar volume, whether in hard copy, on-line, or both.

Steve McConnell. *Code Complete.* Redmond, WA: Microsoft Press, 1993. This is a practical handbook for software construction. McConnell's focus in this book is to advance the common practice of software development to the leading edge. He provides research and programming experience to facilitate development of high-quality software. McConnell provides a chapter entitled Where to Go

for More Information, with descriptions of some of the industry's best books, articles, and organizations. This is recommended reading for any software developer. Although the book predates the Web and client-server technology, most developers I know continue to value the practical and useful advice provided here.

Steve McConnell. "The Power of Process." ***IEEE Computer*** **1998 31(3) pp. 100–102.** Also available at http://www.construx.com/stevemcc/articles/art09.htm. This is a concise, straightforward, clear discussion of the value of using a process. Examples of cutting time-to-market and reducing costs and defects by factors of three to ten are provided from actual organizations. McConnell challenges the view that process is rigid, restrictive, and inefficient. This article is valuable reading for managers of systems and software projects.

Suzanne Robertson and James Robertson. ***Mastering the Requirements Process.*** **Harlow, UK: Addison-Wesley, 1999.** The heart of this book is the Volere Requirements Process Model, a detailed guide. An excellent requirements specification template is available on-line at http://www.atlsysguild.com/GuildSite/Robs/Template.html. Several examples of requirements are provided.

U.S. DoD, Fort Belvior, VA: Defense Standardization Program Office. ***Communicating Requirements.*** **SD-16. 1998.** Available at http://www.dsp.dla.mil. This handbook provides a comprehensive discussion of the requirements process within the DoD, with emphasis on clear, performance-based statements of requirements. The handbook recognizes that a single approach cannot accommodate the varying array of materiel acquisitions. It provides a requirements process, describes the evolution of requirements, provides descriptions of the requirements documents (such as the mission needs statement, operational requirements document, functional description, statement of work [SOW], and system specification), explains the requirements generation flow, describes the analysis supporting requirements determination, provides examples of statements of objectives, presents an acquisition case study, and provides a sample SOW.

Gerald Weinberg. ***Becoming a Technical Leader: An Organic Problem-Solving Approach.*** **New York: Dorset House, 1986.** This is a very readable book that provides good advice based on experience. Weinberg addresses different leadership styles and focuses on the problem-solving style as the one generally utilized by successful technical leaders. This is a recommended desk guide for technical leaders. See also http://www.geraldmweinberg.com/.

Chapter 6

James L. Adams. *Conceptual Blockbusting: A Guide to Better Ideas.* 3rd ed. Reading, MA: Perseus Books, 1986. A stimulus to creative thinking and flexibility, this book is filled with exercises and thoughtful problems that stretch one's mind. It is a great supplement to software design books, aiding algorithm development and the process of partitioning a system into pieces.

Felix Bachmann, Len Bass, Gary Chastek, Patrick Donohoe, and Fabio Peruzzi. *The Architecture Based Design Method.* Technical report CMU/SEI-2000-TR-001, ESC-TR-2000-001. Pittsburg, PA: SEI, 2000. The ABD method fulfills functional, quality, and business requirements at a level of abstraction that allows for variation in producing products. The method provides a series of steps to organize functions, to identify synchronization points for independent threads of control, and to allocate functions to processors. See http://www.sei.cmu.edu/publications/documents/00.reports/00tr001.html.

Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice.* Reading, MA: Addison-Wesley, 1998. Drawing on their experience building and evaluating architectures, Bass and colleagues introduce the concepts and practices concerning how a system is designed and how the system's components interact with each other. Several case studies undertaken with the SEI are provided to illustrate real-world constraints and opportunities. The authors discuss methods for analyzing architectures for quality attributes and provide a good discussion of architecture reviews.

Derek Hatley, Peter Hruschka, and Imtiaz Pirbhai. *Process for System Architecture and Requirements Engineering.* New York: Dorset House, 2000. This book is really about concepts and systems development, especially those involving multiple disciplines. It provides a framework for modeling systems, an architectural model, and a requirements model. The requirements model consists of three submodels (the entity model, process model, and control model) and their supporting specifications. There is a good explanation of the role of the system architect/system engineer (pp. 200–201). The authors use case studies of a hospital's patient-monitoring system and of a multidisciplinary groundwater analysis system to illustrate their principles. An appendix describes misconceptions of the Hatley/Pirbhai methods.

Christine Hofmeister, Robert Nord, and Dilip Soni. *Applied Software Architecture.* Reading, MA: Addison-Wesley, 2000. This book is another in Addison-

Wesley's Object Technology Series. It provides practical guidelines and techniques for producing quality designs, focusing on four views—conceptual, module, execution, and code. Part III presents four architectures developed by the authors at Siemens Corporate Research in Princeton, New Jersey. Hofmeister and colleagues provide examples of what goes into the architecture, the engineering concerns addressed, and how notation is used to describe it. Design trade-offs made by the different architects to solve architectural issues are presented.

W. H. Inmon, John A. Zachman, and Jonathan G. Geiger. ***Data Stores, Data Warehousing, and the Zachman Framework: Managing Enterprise Knowledge.*** **New York: McGraw Hill, 1997.** One of the keys to success for modern corporations is access to the right information at the right time at the right place in the right form. The Zachman Framework was formally published in 1987 to describe an architecture for capturing the aspects of an information system. The Zachman Framework is a model that major organizations can use to view and communicate their enterprise information infrastructure. One of the major applications of the Zachman Framework is to help companies to migrate from legacy systems. It helps companies attain knowledge so that they can be more responsive to change and better poised to compete.

Ivar Jacobson, Martin Griss, and Patrick Jonsson. ***Software Reuse: Architecture Process and Organization for Business Success.*** **New York: ACM Press, 1997.** The authors' vision is that this book will facilitate the practice of object-oriented component-based software engineering. Their belief is that systematic, large-scale reuse, coupled with object technology, is the only way to improve radically the process of software development. Substantial degrees of reuse can be achieved only by radically changing traditional software architectures and development processes. Jacobson and colleagues emphasize that object technology does not yield reuse automatically. An explicit reuse agenda and a systematic approach to design and process are required to achieve a high level of reuse. They provide a framework called the Reuse-Driven Software Engineering Business (RSEB). The work is based on Jacobson's **use case-driven** architecture and process modeling. An appendix addresses the use of the Unified Modeling Language in the RSEB.

Paul Kaminski. ***Reducing Life Cycle Costs for New and Fielded Systems.*** **Office of the Deputy Undersecretary of Defense for Aquisition Reform. December 4, 1995.** This is a memorandum for the secretaries of the military department and others that describes the DoD policy and strategy to develop and field affordable

weapons systems. It includes the CAIV Working Group paper that describes the CAIV approach. See also http://www.acq.osd.mil/ar/.

Rick Kazman and S. Jeromy Carriere. "Playing Detective: Reconstructing Software Architecture from Available Evidence." ***Automated Software Engineering*****, 1999;6:107–138.** New systems development efforts are often constrained by existing legacy applications. Analysts need to be able to extract information from existing systems to use to develop architectures. This paper presents Dali—an open, lightweight **workbench**[18]—that aids an analyst in extracting, manipulating, and interpreting architectural information. Kazman and Carriere emphasize that no tool is right for all jobs and that no extraction technique is useful without user interaction.

Henry Petroski. ***To Engineer Is Human.*** **New York: St. Martin's Press, 1992.** Petroski provides insights into engineering failures. He believes that understanding failure is central to understanding engineering, because engineering design has as its objective the obviation of failure. This is an interesting and important construct in the context of systems and software engineering.

Eberhardt Rechtin and Mark W. Maier. ***The Art of System Architecting.*** **New York: CRC Press, 1997.** Rechtin and Maier provide a table with almost 200 **heuristics** for systems-level architecting, providing access to the underpinnings of principal design guidelines. For example, "a system will develop and evolve much more rapidly if there are stable intermediate forms than if there are not." They note that software is rapidly becoming the centerpiece of complex system design, in the sense that an increasing fraction of system performance and complexity is captured in software. Rechtin and Maier discuss both the architecting of software and the impact of software on system architecting.

Eberhardt Rechtin. ***Systems Architecting of Organizations: Why Eagles Can't Swim.*** **New York: CRC Press, 1999.** Rechtin addresses this book to the challenge of maintaining organizational survival and excellence while adjusting to the new world of global communications, transportation, economics, and multinational security. He identifies factors that can lead to excellence in one field or period of time, but to potential weaknesses in another, and offers insights to address these factors.

Software Engineering Institute. ***The Architecture Tradeoff Analysis Method.*** **1999.** Available at http://www.sei.cmu.edu/activities/ata/ATAM/tsld004.htm. The purpose of the Architecture Tradeoff Analysis Method is to assess the consequences

of architectural decision alternatives in light of quality attribute requirements.

The Chief Information Officers (CIO) Council. ***Federal Enterprise Architectural Framework.*** **Version 1.1. September 1999.** Available at http://www.itpolicy.gsa.gov/mke/archplus/archhome.htm. This framework was developed beginning in April 1998 to promote shared development for common U.S. government processes, **interoperability** and sharing of information among government agencies and other entities. The Clinger-Cohen Act of 1996 assigned CIOs with the responsibility to develop IT architectures. The framework consists of approaches, models, and definitions for communicating the overall organization and relationships of architecture components required for developing and maintaining a Federal Enterprise Architecture. It utilizes the National Institute of Standards and Technology Enterprise Architecture Model.

The Open Group. ***The Open Group's Architectural Framework*** **(*TOGAF*).** Available at http://www.opengroup.org/public/arch/. TOGAF is a tool for defining an IT architecture. TOGAF was developed by The Open Group's own members, working within the TOGAF Program. The original development of TOGAF in 1995 was based on the Technical Architecture Framework for Information Management (TAFIM), developed by the U.S. DoD. The DoD gave The Open Group explicit permission and encouragement to create TOGAF by building on the TAFIM, which itself was the result of many years of development effort and many millions of dollars of U.S. government investment. Starting from this foundation, the members of The Open Group's TOGAF Program developed successive versions of TOGAF in subsequent years and published each version on The Open Group's public Web site. If you are new to the field of IT architecture and/or TOGAF, you may find it worthwhile to read the set of frequently asked questions at this Web site. Here you will find answers to questions such as what is an architectural framework and what are the benefits to an organization by using TOGAF.

Chapter 7

Michael Brassard and Diane Ritter. ***The Memory Jogger II: A Pocket Guide of Tools for Continuous Improvement & Effective Planning.*** **Salem, NH: GOAL/QPC, 1994.** Also available at http://www.goalqpc.com. This pocket-size book is useful for process engineers and others involved in QI. It provides concise summaries of quality tools including the Gantt chart, control chart, flowchart, activ-

ity network diagram, check sheet, force field analysis, prioritization matrices, run chart, and scatter diagram. A tool selector chart that organizes the tools according to typical improvement situations is provided.

Jo Condrill and Bennie Bough. *101 Ways to Improve Your Communication Skills.* Alexandria, VA: Goal Minds, 1998. This book provides straightforward techniques to facilitate communication. It discusses *mind mapping*—a system of recording thoughts so that we employ both left-brain and right-brain thinking (p. 35). Condrill and Bough provide advice for speaking and writing, with appropriate emphasis on behavioral topics. They also provide a great list of sources for further reading (pp. 106–107).

Larry Constantine. *Constantine on Peopleware.* Englewood Cliffs, NJ: Prentice-Hall, 1995. Constantine provides insightful ideas concerning human issues in software development, including quality and productivity, teamwork, group dynamics, project management and organizational issues, interface design, human-machine interaction, cognition, and psychology. The book includes 30 articles. Among the topics discussed are group development (decisions, roles, space, time management), cowboys and cowgirls (teams and mavericks), work organization (seven different models), tools and methods (computer-aided software engineering, modeling, human-computer interface, methods), process improvements (visibility, reuse, just in time, quality), software usability (consistency/conventions, complexity, scope creep, languages, usability, objects), and brave new software (interfaces, wizards, future faces).

Michael Doyle and David Straus. *How to Make Meetings Work.* East Rutherford, NJ: Berkeley Publishing, 1993. Doyle and Straus observe that most organizations spend between 7% and 15% of their personnel budgets on meetings (this does not include time spent preparing for meetings or attending training programs or conferences). *Time spent attending a meeting is time taken away from other opportunities!* Doyle and Straus provide a set of tools and techniques to make groups more effective. They advocate the interaction method, which rests on four well-defined roles: the facilitator, the recorder, the group member, and the chairperson. They describe each of these roles and assert that 7 to 15 people is the ideal size for a problem-solving, decision-making meeting. Everyone should know what to expect before coming to a meeting. Doyle and Straus discuss how to make a presentation.

Roger Fisher and Scott Brown. ***Getting Together: Building Relationships As We Negotiate.*** **New York: Penguin Books, 1988.** Fisher and Brown provide a set of steps that address initiating, negotiating, and sustaining enduring relationships. A strong message is that each of us can make any relationship better if we make the choice to do so.

Milo O. Frank. ***How to Run a Successful Meeting in Half the Time.*** **New York: Simon and Schuster, 1989.** Frank provides suggestions for all aspects of meetings, offering ideas that will certainly be valuable if applied. We know from our experience that we waste a lot of time in meetings. Why not review these suggestions and make some improvements? Although this book is out of print, it is easily available through second-hand bookshops and the popular electronic booksellers. Meetings can be energizing, productive, and satisfying. Learn how to make them this way, and apply these suggestions to your daily work.

Steven Gaffney. ***The Fish Isn't Sick . . . The Water Is Dirty.*** **Training seminar.** Available at http://www.stevengaffney.com. This proactive one-day seminar teaches one how to clean up the communication water and establish honest, effective communication with anyone. Gaffney provides a process that has worked every time I've taken the opportunity to use it. He emphasizes the value of acknowledging the other people involved in our lives.

Charles Handy. ***Gods of Management: The Changing Work of Organizations.*** **New York: Oxford University Press, 1996.** This is an American edition of a book the author wrote in 1978 in England. It provides an insightful view of leadership styles and corporate cultures.

Watts S. Humphrey. ***Why Don't They Practice What We Preach?*** **1998.** Available at http://www.sei.cmu.edu/publications/articles/sources/practice.preach/index.html. I recommend this article to you for insights concerning why technical people do not use improved methods, even when there is clear evidence that the methods help and there is strong pressure to use them. This seems to be true regardless of the engineer's experience and training. Engineers tend to revert to their established ad hoc and informal practices. Only when they are convinced that a method works by seeing results will they even try a new method. Today's organizations have few role models that consistently demonstrate effective work habits and disciplines. This factor accounts for some of the reasons that industry results have not improved in spite of dramatic improvements in practices, methods, techniques, and tools. Practitioners are advised to read and reflect on Humphrey's

insights so that we can develop ways to overcome our failure to take advantage of improvements. McConnell's *After the Gold Rush: Creating a True Profession of Software Engineering* is full of ideas and suggestions to help with this situation. McConnell notes in his epilogue that common development problems won't be avoided without our support.[8]

Otto Kroeger and Janet M. Thuesen (contributor). *Type Talk at Work: How the 16 Personality Types Determine Your Success on the Job.* New York: Dell Publishing, 1993. The authors explain how managers, executives, and workers can evaluate personality types and achieve improved job effectiveness. They provide suggestions for how to deal with individuals who are opposite of your type.

Six Sigma Qualtec. *QI Story: Tools and Techniques, A Guidebook to Problem Solving.* 3rd ed. Tempe, AZ: Six Sigma Qualtec, 1999. This tiny reference book provides a concise and helpful description of the concepts of total quality management and an overview of the seven-step QI story. It also includes summaries of QI tools and techniques such as brainstorming, multivoting, the Pareto chart, the Ishikawa (fishbone) diagram, countermeasures (solutions), cost-benefit analysis, barriers-and-aids analysis, graphs, histograms, process flowcharts, and control charts. Available from Six Sigma Qualtec, 480-586-2600.

Douglas Stone, Bruce Patton, and Sheila Heen. *Difficult Conversations: How to Discuss What Matters Most.* New York: Penguin Books, 1999. It's natural to avoid conversations that cause anxiety and frustration. This book provides an approach for having difficult conversations with less stress and more success. We all bring erroneous but deeply ingrained assumptions into our daily activities. This book provides valuable insights for anyone who works with others.

Gerald M. Weinberg. *Quality Software Management: Congruent Action.* Vol. 3. New York: Dorset House, 1994. This book deals with the ability to act appropriately in difficult interpersonal situations—an essential ability for successful software development managers. Weinberg uses simple but effective models to explain human behavior, and he uses examples from the software engineering industry to put these models in contexts familiar to software developers. He draws on his 40 years of work in the industry to discuss various styles of coping (especially under stress), selection of managers, the importance of self-esteem, how to transform incongruent behavior into effective actions, addictive behaviors, and how to create and manage productive teams. He addresses the important question of why people do things wrong when they know how to do them right.

Neal Whitten. ***Becoming an Indispensable Employee in a Disposable World.*** **Amsterdam: Pfeiffer & Company, 1995.** The author emphasizes the value of capitalizing on key personal traits, such as self-esteem and communication, noting that we mirror our self-expectations! He provides a step-by-step process to becoming a self-directed employee. He also addresses balancing our professional and personal lives. Recommended reading for everyone.

Chapter 8

Barbara A. Bicknell and Kris D. Bicknell. ***The Road Map to Repeatable Success: Using QFD to Implement Change.*** **Boca Raton, FL: CRC Press, 1995.** The authors are associated with Bicknell Consulting, Inc., and have extensive experience applying the QFD methodology. Their book is a very comprehensive treatment of QFD, explaining not only how QFD can be applied to all levels of the organization, but also providing detailed guidance concerning how to create a QFD matrix, analyze it, and develop an integrated plan using it. They provide a nine-step approach to developing and using a QFD approach. A useful chapter concerning how to develop a pilot QFD program is provided. Several case studies and examples are provided.

Ian Graham. ***Requirements Engineering and Rapid Development: An Object-Oriented Approach.*** **Reading, MA: Addison-Wesley, 1998.** Graham has responsibility for software methods at Chase Manhattan Bank and believes that development should be done quickly and effectively. He provides practical advice concerning object modeling techniques. His approach complies with the principles of the Dynamic Systems Development Method that was developed by a consortium of 17 users in England. This method does not include or recommend techniques. See J. Stapleton, *Dynamic Systems Development Method: The Method in Practice,* Harlow, UK: Addison-Wesley, 1997.

IFPUG. ***Function Point Counting Manual.*** **Current release. Westerville, OH.** The use of function points as a measure of the functional size of information systems has grown rapidly. Function points are a more objective measure of software development than lines of code from the customer's point of view because of the characteristics of particular software languages. Recent releases of this manual have provided a consensus view of the rules of function point counting (IFPUG standard). The manual provides a discussion of the objectives and bene-

fits of FPA, an overview of FPA, and function point counting procedures. Examples are provided to explain function point counting practices, concepts, rules, and procedures. Complementary IFPUG documentation is available, including an IFPUG brochure (with membership application), *Guidelines for Software Measurement, Application of Measurement Information, Quick Reference Counting Guide, Function Point Analysis Case Studies,* and IFPUG glossary. A list of instructors providing certified training courses is also available.

Michael Jackson. *Software Requirements & Specifications.* Wokingham, UK: Addison-Wesley, 1995. Jackson describes a large set of methods, including data flow diagrams, entity relationship span, frame diagrams, graphic notations, tree diagrams, and the top-down approach. He shares a set of principles and prejudices based on his 30 years in software development. His other books include *Principles of Program Design* (Academic Press, 1975) and *System Development* (Prentice Hall International, 1983).

Capers Jones. *Estimating Software Costs.* New York: McGraw Hill, 1998. This is a very thorough treatment of this subject. Jones worked for IBM and is founder and chairman of Artemis-SPR, Inc. (see http://www.spr.com). He has been collecting historical data and designing and building software cost estimating tools since 1971. Jones's awareness that there are hundreds of factors that determine the outcome of a project and his extensive database enable him to advise others concerning the factors that are the most critical.

Capers Jones. *Positive and Negative Factors That Influence Software Productivity.* Version 2.0. Burlington, MA: Software Productivity Research, Inc., 1998. This is an extremely insightful paper. Jones notes that software productivity is complex, with at least 100 known factors that can influence the outcome of software projects. The conclusions are derived from 2,000 software projects examined between 1993 and 1998. The requirements activity comprises an average of 8.42% of the total effort. Jones provides lists of factors that exert both positive and negative impacts on software productivity. The cumulative results of negative factors are much larger than those of the positive factors. This means that it is easier to make mistakes and degrade productivity than it is to get things right and improve productivity. Jones also provides data concerning the impact of positive and negative factors on maintenance productivity. He addresses "best in class" and "worst in class" companies. The most common pattern noted for both systems and software domains is projects and companies in which the technical work of building software (design and coding) is reasonably good, but proj-

ect management factors and quality control factors are fairly weak.

James Martin. *Rapid Application Development.* New York: Macmillan Publishing, 1991. Martin provides a comprehensive treatment of RAD. He describes a requirements planning phase called *joint requirements planning* (JRP). JRP utilizes a workshop to examine goals, problems, critical success factors, and strategic opportunities to determine system objectives, departments and locations served, determination and prioritization of system functions, process flow, and a list of unresolved issues. The basic idea of JRP and JAD techniques is to select key end users and to conduct workshops that progress through a structured set of steps for planning and designing a system. This book is recommended reading for individuals considering the use of RAD techniques. Martin also provides an excellent discussion of metrics, tools, methodology, people, and management, as well as techniques including prototyping, data modeling, and others.

Mark C. Paulk. *A Comparison of ISO 9001 and the Capability Maturity Model for Software.* Technical report CMU/SEI 94-TR-12. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, July 1994. This report provides a detailed mapping between International Standard Organization (ISO) 9001 and the Capability Maturity Model (CMM). Paulk concludes that there is a strong correlation between ISO 9001 and the CMM, although some issues in ISO 9001 are not covered in the CMM, and some issues in the CMM are not addressed in ISO 9001. He believes that the biggest difference between the two standards is the emphasis of the CMM on continuous process improvement. ISO 9001 addresses the minimum criteria for an acceptable quality system. Both documents emphasize processes that are documented and are practiced as documented. However, Paulk concludes that a CMM level 1 organization could be certified as compliant with ISO 9001. If an organization is following the spirit of 9001, it seems probable the organization would be near or above CMM level 2. A CMM level 2 organization would have little difficulty in obtaining ISO 9001 certification.

Jack B. Revelle, John W. Moran, and Charles Cox. *The QFD Handbook.* New York: John Wiley & Sons, 1997. QFD uses a number of matrices that translate customer requirements into engineering or design requirements. Revelle and colleagues apply QFD to several areas such as ISO 9000, service design, and software design. A disk that supplies the QFD Pathway software tool package is packaged with the book.

Linda Rosenberg. *Writing High Quality Requirement Specifications.* Tutorial presented at the 12th International Software Quality Week (QW'99). San Jose, CA. May 24–28, 1999. The requirements specification establishes the basis for all of the project's engineering, management, and quality assurance functions. If the quality of the requirements specification is poor, it can create risks in all areas of the project. The tutorial addresses effective development of quality requirement specifications and provides ideas and methods that can be incorporated into the project plan. These produce a return on documentation effort and improved comprehension. See http://www.soft.com/QualWeek/QW99/qw99.abs.html.

John Terninko. *Step-by-Step QFD: Customer-Driven Product Design* 2nd ed. Boca Raton, FL: St. Lucie Press, 1997. This is an excellent book that describes why to use QFD as a technique and provides a step-by-step guide for how to use it. The author provides suggested workshops and sample worksheets. An essential point is noted on page 3: "Once customer needs are understood . . ." In other words, one must know the *real* requirements prior to taking advantage of the QFD technique. Assuming use of the real requirements as the base, one can achieve reduced development time by a factor of 2 or 3 (that is, 1/2 to 1/3). The author notes that implementing QFD for new product designs requires a substantial initial investment of resources. Unfortunately, the book does not provide a system development or software development example. Rather, the examples of the application of QFD are limited to manufacturing. The book does discuss the origin and application of TRIZ, a Russian acronym that translates to Theory of Inventive Problem Solving (TIPS).

Jefffrey L. Whitten, Lonnie D. Bentley, and Kevin C. Dittman. *Systems Analysis and Design Methods.* 5th ed. Boston, MA: McGraw-Hill, 2000. This popular textbook provides a comprehensive discussion of most information systems development topics. I particularly like the insight that classic, structured, and "modern" techniques are (or should be) mutually supportive, not mutually exclusive. The definitions of terms in the index help clarify an understanding of topics. The volume incorporates an adaptation of Zachman's *Framework for Information Systems Architecture* (color mappings to data, processes, geography, interfaces, and objects) so that in reviewing the figures, one can discern which is which. For each chapter Whitten and colleagues provide a set of suggested readings that seem to be well thought through.

Neal Whitten. "Meet Minimum Requirements: Anything More Is Too Much." PM Network (September 1998), p. 19. See also http://www.pmi.org. Whitten

advocates committing to a project plan that includes only essential function, with a "closet plan" for nonessential function. Deliberately practicing meeting minimum requirements helps an organization or company be first-to-market, earn increasing credibility from its clients, and strongly posture its enterprise for taking on new business opportunities. USA. Phone: (610) 356-4600. Fax: (610) 356-4647.

Chapter 9

Barry W. Boehm. *Software Engineering Economics.* Englewood Cliffs, NJ: Prentice Hall, 1981. Boehm's book is a classic that identifies the factors most strongly influencing software costs and provides methods to determine the estimated costs of a software project. He presents the Constructive Cost Model and provides case study examples of its use. The book has an excellent section concerning people-related reasons for variability in software estimation (pp. 666–676).

Elfriede Dustin, Jeff Rashka, and John Paul. *Automated Software Testing: Introduction, Management, and Performance.* Boston, MA: Addison-Wesley, 1999. This book is a comprehensive, step-by-step guide to the most effective tools, techniques, and methods for automated testing. The Automated Test Lifecycle Methodology (ATLM) is a structured process for designing and executing testing that parallels the rapid application development methodology. The book provides guidance on all aspects of the testing process. A compact disk comes with the book that contains ATLM graphics in PDF, JPEG, EPS, and TIFF formats. There is extensive discussion of requirements-related testing topics.

Jeffrey O. Grady. *System Validation and Verification.* Boca Raton, FL: CRC Press, 1997. This book covers all aspects of V&V. Grady has extensive experience and provides practical methods for each aspect of the topics. The book is extensively illustrated with helpful explanatory figures.

Theodore F. Hammer, Leonore L. Huffman, and Linda Rosenberg. "Doing Requirements Right the First Time." *CrossTalk,* 1998:20–25. Hammer et al. address three critical aspects of requirements: definition, verification, and management. Project data collected from the National Aeronautics and Space Administration's Goddard Space Flight Center by the Software Assurance Technology Center (SATC) are used to demonstrate key concepts and to explain how to apply them to any project. SATC's Automated Requirements Measurement

tool is used, and seven measures are developed (lines of text, imperatives, continuances, directives, weak phases, incomplete, and options). These metrics provide insight into the completeness of the test program and an understanding of the characteristics of the verification program.

James D. Palmer. "Traceability." In: R. H. Thayer and M. Dortman, eds. *Software Requirements Engineering.* Los Alamitos, CA: IEEE Computer Society Press, 1997: 364–374. Palmer shows that traceability gives essential assistance in understanding the relationships that exist within and across requirements, design, and implementation. He points out that traceability is often misunderstood, frequently misapplied, and seldom performed correctly. This work is recommended reading for requirements engineers.

William Perry. *Effective Methods for SoftwareTesting.* New York: John Wiley & Sons, 1995. Perry provides extremely helpful guidance concerning verification and testing activities that need to accompany the problem definition and requirements analysis activities in a development effort. He emphasizes that failure to do this will result in much higher testing costs later in the project. He provides a detailed discussion of *requirements phase testing,* including recommendations for test tools (Walk-Through and Risk Matrix) and an extensive listing of application risks. Because testing during the requirements phase is a new concept to many development teams, Perry provides a requirements phase test process.

Delores R. Wallace and Laura M. Ippolito. "Verifying and Validating Software Requirements Specifications." In: Richard H. Thayer and Merlin Dortman, eds. *Software Requirements Engineering.* 2nd ed. Los Alamitos, CA: IEEE Computer Society Press, 1997: 389–404. Wallace and Ippolito describe V&V activities and emphasize that V&V is a powerful tool for improving intermediate products such as requirements specifications, design descriptions, test cases, and test procedures. They provide descriptions of 28 test techniques and suggest strategies for choosing among them. A rich set of references is provided.

Chapter 10

Mark C. Paulk, Charles V. Weber, Suzanne M. Garcia, Mary Beth Chrissis, and Marilyn Bush. *Key Practices of the Capability Maturity Model.* Version 1.1. Pittsburgh, PA: Software Engineering Institute, Carnegie-Mellon University, 1993. Also available at http://www.sei.cmu.edu/publications/documents/93.

reports/93.tr.025.html. Many don't realize that the SW-CMM, version 1.1, is only 64 pages in length. *Key Practices of the CMM,* version 1.1, is much more extensive and describes the practices for each of the 18 key process areas. These are organized according to a set of common features: commitment to perform, ability to perform, activities performed, measurement and analysis, and verifying implementation. I have been applying these practices to projects and organizations for 12 years and have found them very helpful in improving the development process.

Daniel P. Petrozzo. *The Fast Forward MBA in Technology Management.* New York: John Wiley & Sons, 1998. This book is one from the Fast Forward MBA Series. Its aim is to facilitate the use of technology in an organization. Petrozzo provides examples of companies taking advantage of leading technologies and describes how to manage technology. An interesting section is provided concerning changing requirements. Two pitfalls identified by Petrozzo are (1) too large of an initial implementation and (2) dependence on outside providers of software, hardware, and expertise.

Preston G. Smith and Donald G. Reinertsen. *Developing Products in Half the Time.* 2nd ed. New York: John Wiley & Sons, 1998. This book provides useful concepts, methods, and metrics for reducing the time required to develop products. In this new edition, Smith and Reinertsen have drawn on their experience in working with clients to provide practical tools to accelerate the development process. For example, they stress the value of partnering in off-site workshops to create specifications jointly.

Ian Sommerville. *Software Engineering.* 5th ed. Reading, MA: Addison-Wesley, 1995. Sommerville is concerned that although there has been tremendous progress in software engineering, there has been a relatively slow diffusion of this progress into industrial practice. He perceives a need to transfer proven practices into everyday use. An extensive treatment of requirements and specifications is provided, with emphasis on a requirements engineering process. An emphasis on prototyping as being useful in validating the systems requirements is provided.

Ronald Starbuck. "How to Control Software Changes." *Software Testing and Quality Engineering (STQE) Magazine* 1/6 1999:18–21. Starbuck provides a high-level change control process, describes the factors for how change requests are evaluated and explains the role of a CCB. He considers the business environment and the people involved and stresses the need for sponsorship. CCB infrastructure sup-

port includes policy (guiding principles), process (a flowchart), procedures (explaining how to accomplish the process steps), and authorization (agreement on what you are or are not empowered to do or not to do). The configuration management plan (CMP) defines how to use the process and procedures in the specific lifecycle situation, indicating who uses configuration management, what work products they use it on, and when they use it. An excellent, easily tailorable model for a CMP is *IEEE Standard for Software Configuration Management Plans* (IEEE Std 828, 1990).

Gerald M. Weinberg. "Just Say No! Improving the Requirements Process." *American Programmer* (10) 1995:19–23. Weinberg's view is that for many organizations, the principal barrier to higher quality is an inadequate requirements process. He suggests a four-step process: (1) measure the true cost and value of requirements, (2) gain control of the requirements inputs, (3) gain control of the requirements outputs, and (4) gain control of the requirements process itself.

Chapter 11

ABT Corporation. *Core Competencies for Project Managers.* 2000. Available at http://www.tsepm.com/may00/art5.htm. The authors assert that core competencies of PMs should be divided into soft and hard skills. The soft skills (based on years of feedback from customers) include visible leadership, flexibility, sound business judgment, trustworthiness, possession of several effective communication styles, ability to act as a coach and mentor, active listening skills, ability to set and to manage expectations, ability to provide constructive project negotiations, ability to facilitate issue and conflict resolution, and ability to provide organizational and leadership skills. The hard skills include project definition, planning, and estimating and providing a control process.

Eliyahu M. Goldratt. *Critical Chain.* Great Barrington, MA: The North River Press, 1997. This is a business novel that introduces you to Goldratt's thinking processes. It is thought provoking and stimulating.

Eliyahu M. Goldratt and Jeff Cox. *The Goal.* 2nd rev. ed. Great Barrington, MA: The North River Press, 1992. This is also a business novel considered by many to be a very important business book. It introduces the Theory of Constraints and emphasizes eliminating bottlenecks.

Robert Grady. ***Practical Software Metrics for Project Management and Process Improvement.*** **Englewood Cliffs, NJ: Prentice Hall, 1992.** This second book by Grady extends the concepts and examples in his first book based on the design and implementation of a software metrics program at Hewlett Packard. Grady states that if you are a PM or if you are involved in process improvement, this book is for you. Grady believes that the SEI CMM is a model that will help products move toward continuous process improvement—a key to our future. He also believes that a lot depends on what we believe we can do. He recommends that we use the techniques and ideas in the CMM, apply them to our projects, and use them to set continuous improvement goals for our project teams. He believes that we will be surprised at what it is possible to accomplish.

Robert Grady and D. Caswell. ***Software Metrics: Establishing a Companywide Program.*** **Englewood Cliffs, NJ: Prentice Hall, 1987.** This book provides the history, mechanics, and lessons learned from the example of the Hewlett Packard company's creation, design, development, and implementation of a successful software metrics program. Hewlett Packard, through its Software Metrics Council, determined to collect size, effort, schedule, and defects data initially. Grady and Caswell emphasize that the greatest benefits of collecting metrics are experienced by PMs through better understanding of the process that their team is following and through measurable indicators of project status.

Capers Jones. ***Software Quality in 2000: What Works and What Doesn't.*** **January 18, 2000.** Briefing available from Software Productivity Research, Inc., at http://www.spr.com. This is a comprehensive briefing based on the database of software projects maintained by Software Productivity Research, Inc., that includes 600 companies, 30 government and military groups, roughly 9,000 total projects, and data from 24 countries. Jones identifies practices that provide good-, mixed-, and minimal-quality results, thus suggesting approaches that provide the best return on the cost and effort invested.

Craig Kaplan, Ralph Clark, and Victor Tang. ***Secrets of Software Quality.*** **New York: McGraw-Hill, 1995.** Kaplan and colleagues report 40 innovations from IBM that address culture, leadership, process, and tools. There are many excellent suggestions for use in a forward-looking organization. A quality maturity assessment method that is based on the 1994 Malcolm Baldrige Quality Award criteria is provided.

Steve McConnell. *Software Project Survival Guide,* Redmond, WA: Microsoft Press, 1998. This is my favorite reference for providing advice, suggestions, and practical help for a systems or software project. McConnell provides a project survival test that gives insight into requirements, planning, project control, risk management, and personnel issues. He then proceeds to provide useful suggestions and survival checks in each area. One can't help being helped by this book. See also http://www.construx.com/stevemcc/.

Fergus O'Connell. *How to Run Successful Projects II—The Silver Bullet.* 2nd ed. New York: Prentice Hall, 1996. O'Connell is a principal of ETP, The Structured Project Management Company, in Ireland. He presents a straightforward approach for project management consisting of ten steps, with emphasis on project planning and tracking and use of an automated project-tracking tool such as Microsoft Project. He presented this approach in the first edition of his book and then tried it with the companies with which he was consulting—it worked! Hence the unfortunate subtitle *The Silver Bullet*. O'Connell uses a PSI indicator to evaluate the status and probability of success of a project. This is highly recommended reading for every PM and anyone with management responsibilities. See http://www.etpint.com/.

Lawrence H. Putnam and Ware Myers. *Measures for Excellence: Reliable Software on Time, Within Budget.* Upper Saddle River, NJ: Yourdon Press, 1992. Putnam and Myers emphasize the life cycle model and provide a simple software estimating system. They provide a glossary of more than 100 terms used in quantitative software management. They explain how conceptual work like software development has been found to progress according to a mathematical curve known as the *Rayleigh distribution*. This formula helps to understand what happens when you compress a schedule, estimate new projects, and add people to a late project, as well as to be able to project the number of defects remaining in a work product. Study of this book facilitates understanding of reliability.

Walker Royce. *Software Project Management: A Unified Framework.* Reading, MA: Addison-Wesley, 1998. An excellent read. "Key Points" are provided at the beginning of each chapter, summarizing the main themes. Royce gives attention to project economics, including improving processes and team effectiveness. He provides a management process framework, with emphasis on iterative process planning. He suggests seven core metrics and describes tailoring of processes (including an example of a small-scale project versus a large-scale project). Useful

appendixes include The COCOMO Cost Estimation Model, Change Metrics, CCPDS-R Case Study, and Process Improvement and Mapping to the CMM.

Rob Thomsett. *Third Wave Project Management.* Upper Saddle River, NJ: Yourdon Press, 1993. This is a useful handbook that encourages use of updated management ideas and techniques for project initiation, planning, estimation, and risk assessment. Thomsett believes that the emergence of new development techniques such as joint requirements planning, joint application design, and rapid application development require a more dynamic and real-time project management approach than is typically used. They require a new focus on team formation, structure, and management, based on pressures for increased productivity, fewer people, and more client-oriented service.

Bruce F. Webster. *Pitfalls of Object-Oriented Development.* New York: M&T Books, 1995. This is a superb book that describes pitfalls, not only for OO development but for development in general. Webster candidly shares his wealth of experience, describing each pitfall and noting symptoms, consequences, detection, extraction, and prevention for each one. This is a valuable read for any developer or manager.

Ed Weller. "Practical Applications of Statistical Process Control." In: *Proceedings of the 10th International Conference on Software Quality.* This is an excellent article that explains how to use SPC to improve project success. Applying quantitative methods and SPC to development projects can provide a positive cost-benefit return. Quantitative analysis of inspection and test data is used to analyze product quality during test and to predict postship product quality for a major release. The processes used, decisions made using the project's data, and the results obtained are described. Weller advises that the following questions be asked about any metric or analysis technique: (1) Is it useful, and does it provide information that helps make decisions? (2) Is it usable, and can we reasonably collect the data and do the analysis?

Neal Whitten. *Managing Software Development Projects: Formula for Success.* 2nd ed. New York: John Wiley & Sons, 1995. Whitten's focus is on practical, easy-to-implement solutions to common problems he has found in consulting with organizations. Formerly with IBM, Whitten has managed a variety of projects. Among the areas addressed are personnel, quality, project scheduling and tracking, product requirements, and product quality and usability. Managers will find many helpful lessons.

Chapter 12

Frederick P. Brooks, Jr. *The Mythical Man-Month.* Anniversary ed. Reading, MA: Addison-Wesley, 1995. This is probably the most cited software project management book of all time. In the first edition of this book (1975), Brooks drew on his experience as the project manager for the IBM/360 computer family and then for OS/360 to provide propositions concerning software engineering and programming. One assertion was that the man-month is mythical as a measure of productivity. In his twentieth anniversary edition (1995) of this book, Brooks notes that he was struck by how few of the propositions have been critiqued, proved, or disproved by ongoing software engineering research and experience! Chapter 16 reprints "No Silver Bullet: Essence and Accidents of Software Engineering," a 1986 paper that was reprinted in 1987 in IEEE *Computer* magazine. This paper predicted that the next decade would not see any single development in either technology or management technique that by itself would provide even one order of magnitude of improvement in productivity, reliability, or simplicity. Brooks's position 20 years later is still that software development is difficult.

Alan M. Davis. *Software Requirements: Objects, Functions, and States.* Rev. Upper Saddle River, NJ: Prentice Hall PTR, 1993. This book provides a good discussion of "what is a requirement?" It also provides a survey of techniques, including object-oriented problem analysis, function-oriented problem analysis using data flow diagrams, and state-oriented problem analysis. It has a detailed discussion of the attributes of a software requirements specification and a very exhaustive (772 sources) annotated bibliography as of 1993.

James A. Highsmith III. *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems.* New York: Dorset House, 1999. As characterized by Ken Orr, the message of this book is that large information systems don't have to take so long to develop, they don't have to cost so much, and they don't have to fail. This is hard to disagree with. The recommended solution is a radical form of incremental development. Recommended techniques include using customer focus groups, versioning, **time-boxed** management, and active prototyping in combination. The book is advertised to provide a framework to build independent subprojects in small, short-term pieces. Adaptive cycles are mission driven, component based, iterative, time boxed,[8] risk driven, and change tolerant. Highsmith advocates the *optimization paradox,* basically denying that process improvement (for example) could have any beneficial effect (p. 187). Another theme of the book that is also hard to disagree with is that collaboration

enables organizations to generate emergent results (p. 126). Highsmith asserts that "active partnerships are required" (p. 158). He believes that inspections (encompassing reviews, inspections, and walkthroughs) have the most consistently proven track record of all software engineering techniques implemented since 1980. Chapter 7, Why Even Good Managers Cause Projects to Fail, is insightful.

Watts S. Humphrey. *Introduction to the Personal Software Process.* Reading, MA: Addison-Wesley, 1997. This excellent book provides the background and approach for implementing the Personal Software Process (PSP). The PSP is a methodology based on process improvement principles that enables developers to develop high-quality products consistently and efficiently. The PSP is offered in a course that teaches developers to measure and to manage the quality of their work using defect density, **defect removal** rate, and yield versus productivity to analyze size, time in phase, defects, and schedule measures. Experience has shown that a serious commitment to the PSP is required, that management support is essential, and that transition to practice must be actively managed. Humphrey's earlier book, *A Discipline for Software Engineering,* also addresses the PSP. See also the article by Ferguson and colleagues titled "Results of Applying the Personal Software Process."

Watts S. Humphrey. *Managing Technical People: Innovation, Teamwork, and the Software Process.* Reading, MA: Addison-Wesley, 1997. Humphrey had nearly 50 years of experience working with technical people when he wrote this book. As he notes, history is a marvelous teacher as long as we are willing to learn. The key is to understand and to respect people and to follow sound management principles, applying them with a healthy sprinkling of common sense. This book adds several chapters to Humphrey's earlier book, *Managing for Innovation: Leading Technical People,* including one concerning the process improvement strategy and describing its power. Topics include respect for the individual, motivating technical and professional people professional discipline, developing technical talent, managing innovative teams, and managing change.

Ravi Kalakota and Marcia Robinson. *e-Business: Roadmap for Success.* Reading, MA: Addison-Wesley, 1999. This book facilitates understanding how electronic business (e-business) impacts current business strategies, applications, and models. Kalakota and Robinson have extensive experience in e-commerce and e-business. A thesis is that technology is now a cause and driver in forming business strategy and that e-commerce is enabling organizations to listen to their

customers and to become the cheapest, the most familiar, or the best. An interesting concept is *customer relationship management*—an integrated sales, marketing, and service strategy that depends on coordinated actions.

Mitch Lubars, Colin Potts, and Charles Richter. "A Review of the State of the Practice in Requirements Modeling." In: *Proceedings of IEEE International Symposium on Requirements Engineering.* Los Alamitos, CA: IEEE Computer Society Press, 1993: 2–14. The authors conclude that it is not easy to make specific recommendations concerning how to improve requirements practices. They observe from other studies that accurate problem domain knowledge is critical to the success of a project, and requirements volatility causes major difficulties during development. Lubars and colleagues encountered several cases of customer-generated ("stated") requirements documents that were hundreds of pages long: "But verbosity does not imply clarity of understanding." Many customer-specific projects employ several domain specialists. Their survey indicated that no companies really know how to assign and to modify priorities or how to communicate those priorities effectively to project members. They observed several times that a "small" change to the requirements caused a large change to the design. The customer/developer partnership is preferred to foster interaction and to promote consensus. Few projects used any particular requirements method. In no case did the researchers find a coherent relationship between requirements analysis and project planning. Requirements engineers did not know how their project managers estimated costs or scheduled milestones. An important finding is that software professionals are notoriously undercapitalized relative to professionals in other engineering or manufacturing fields.

Steve McConnell. *After the Gold Rush: Creating a True Profession of Software Engineering.* Redmond, WA: Microsoft Press, 1999. McConnell's easy-to-read style prevails in this excellent analysis of the status of software engineering today and what should be done. He puts the state of current practices into context and notes that each of us has a choice: to stay with "code-and-fix development practices" or to venture boldly toward a true profession. McConnell made his choice years ago, as is evident from his many important contributions, ministering as he does in a practical manner to the needs and welfare of our industry and humankind. It's time for each of us to read this book, digest it, and join him.

Steve McConnell. *Rapid Development.* Redmond, WA: Microsoft Press, 1996. This is another of McConnell's great books—a testimony to why process improvement is never finished! McConnell takes the view that rapid develop-

ment is not a "glitzy methodology," but rather the use of good practices, time, and effort to achieve an effective development process. He advocates: (1) choosing effective practices rather than ineffective ones, and (2) choosing practices that are oriented specifically toward achieving your schedule objectives. After a discussion of the major topics in development (including partnering), McConnell proceeds to discuss 43 best practices. This is recommended reading for anyone involved in the development process.

Fergus O'Connell. ***How to Run Successful High-Tech Project-Based Organizations.*** **Boston, MA: Artech House, 1999.** In this book, O'Connell leverages his six years of success of his company, ETP (Eyes on the Prize), Inc., in helping to change the behavior of the people running projects and organizations. He applies his structured project management approach to organizations. The focus is on causing people to do the things that create results in customer satisfaction, reduced time-to-market, gaining market edge, increased revenue, and increased profits.

Peter Senge, Art Kleiner, Charlotte Roberts, Richard Ross, George Roth, and Bryan Smith. ***The Dance of Change.*** **New York: Doubleday, 1999.** This book is full of ideas concerning organizational change. It provides a vision of growth and prosperity based on the concept of the learning organization. It is a valuable resource for a leader seeking new possibilities.

Paul A. Strassmann. ***The Squandered Computer: Evaluating the Business Alignment of Information Technologies.*** **New Canaan, CT: The Information Economics Press, 1997.** Strassmann believes that U.S. companies are excessively overspending on computers and that there is no demonstrable relationship between computer spending and corporate profits. He points out that overhead costs of U.S. firms have grown faster than revenues or profits, and he feels that computers have not increased worker productivity. He asserts that 31% of computer projects are canceled and that 53% overrun their budgets. Strassmann believes that the computer trade press has a tendency to popularize examples of excellence in computer usage that disregard financial results. He believes that the cyclical investment pattern for computers is as much a reflection in shifts in organizational power as the result of technological innovation. He provides 152 recommendations for what to do (pp. 389–400). This book is clearly not representative of everyday thinking, and for that reason it is provocative and challenges us to evaluate our "typical" approach.

R. H. Thayer and M. Dorfman, eds. *Software Requirements Engineering.* 2nd ed. Los Alamitos, CA: IEEE Computer Society Press, 1997. This is a valuable resource that includes a collection of informative articles on topics including what is a requirement; system and software engineering requirements elicitation techniques, including use cases; requirements methodologies and tools; traceability; requirements and quality management; and life cycle models. Two Institute of Electronic and Electrical Engineers standards (IEEE *Recommended Practice for Software Requirements Specifications* and "Guide for Developing System Requirements Specifications") and a comprehensive software requirements engineering glossary are provided.

Gerald M. Weinberg. *Quality Software Management.* Vol. 4. Anticipating Change. New York: Dorset House Publishing, 1997. This is the fourth and last volume in a series by Weinberg titled *Quality Software Management*. Weinberg's focus in this book is on how to create a supportive environment. His tenet is that management creates the environment in which the development work is performed. Without improving management, spending on methodologies, tools, application packages, and even training won't help. Based on my own experience, it's hard to disagree with this view. There are two chapters that specifically focus on requirements principles and process.

Karl E. Wiegers. *Creating a Software Engineering Culture.* New York: Dorset House Publishing, 1996. This book provides lots of suggestions and ideas based on Wiegers' experience at Eastman Kodak Company. He provides practical approaches to support process improvement and development efforts, and he identifies culture builders and culture killers that are important to consider.