references. And many software practitioners and authors are listed here together with their key works. The list includes Boehm, Brooks, McConnell , Weinberg, DeMarco, Lister, and, of course, Glass himself.

Most of the facts presented are well known. The fallacies presented are few and in a manner similar to the controversies, many of them seem forced. For example: "Fallacy 1: You can't manage what you can't measure." One would think that after listing this as a fallacy, Glass would offer evidence of why this is false; however, all Glass offers is this mild complaint to close this section "...measurement is vitally important to software management, and the fallacy lies in the somewhat cutesy saying we use to capture that." "Fallacy 4: Tools and Techniques: one size fits all." I am not sure who has been propagating this fallacy and why it deserves a place in the book.

The list of facts and fallacies is a useful reminder to all software practitioners and the many books referenced a source of excellent further reading. The often repeated reference to "controversy" and "fallacy" where there is hardly any does not take away from the value of this quick- and easy-reading book.

Component-Based Development: Principles and Planning for Business Systems

Katherine Whitehead. 2002. Boston: Addison-Wesley. 200 pages. ISBN 0-201-67528-5. (CSQE Body of Knowledge area: Software Engineering Processes)

Reviewed by Carolyn Rodda Lincoln Carolyn.Lincoln@titan.com

Component-Based Development is an introduction to the principles and planning of components for business systems. It provides an introduction to the issues surrounding the use of components in new systems and those that are in maintenance mode. The author points out the "sticking points" in using components and then provides suggestions on how to address them.

The book is divided into four parts. Part 1 is an introduction on what components are and why one would want to use them. Part 2 is the process of planning for the use of components, including both technical and organizational issues. Part 3 is how to build and assemble components into a working system. Part 4 is a short case study of a fictional insurance company that is evolving from mainframe to Internet systems. There is also a glossary of component-related terms.

The author assumes that the reader is knowledgeable about systems and technology although not familiar with components. Many technical terms and acronyms are used without explanation or definition, so the book would be inappropriate for a beginner. The author occasionally uses examples to illustrate concepts, but it would have been valuable to have more. The stated purpose of the book is at a high level, that is, systems architecture, but the author does refer to specific implementations such as Enterprise JavaBeans (EJB) and Microsoft's Component Object Model (COM). The vendors and standards are changing so quickly that those specifics will quickly get out of date; however, the principles will still be applicable.

Component-Based Development is a valuable resource for a software development manager or system architect who needs to understand components. The author provides a balanced viewpoint of the pros and cons of both components and implementation issues surrounding components. The section on how to start using component-based development was particularly helpful. Once the decision is made to use components, the practitioners would have to look elsewhere for training on EJB or COM, but they would have a good background on the questions to ask and the issues to address. The book is highly recommended for anyone in the computer field who is familiar with object-oriented concepts and wants to move on to component-based development.

PROGRAM AND PROJECT MANAGEMENT

Building a Project-Driven Enterprise: How to Slash Waste and Boost Profits Through Lean Project Management

Ronald Mascitelli. 2002. Northridge, Calif.: Technology Perspectives. 368 pages. ISBN 0-9662697-1-3. (CSQE Body of Knowledge areas: Program and Project Management)

Reviewed by Dave Zubrow (dz@sei.cmu.edu)

This is a book that is well worth reading for its practical approach to streamlining many project management practices. Readers from the software world will enjoy many of the examples as the author draws upon his background associated with the development of software intensive systems, even though the book is not strictly about software project management. The book is organized into four parts: principles that drive project efficiency, methods of lean project management, the special case of new product development, and building a project-driven enterprise. The chapters in the first part describe a set of principles for achieving a lean approach to running projects. The ideas here are presented in a straightforward manner with examples that help readers understand why the principles are so important. On the second page of the book, the author drives home the focus of the book by presenting a time log of a worker's day. The important point of the time log is the assignment of value to various activities. Out of an 8.5 hour workday, only 1.5 hours actually generated value. While this may seem extreme, data from TSP^sm teams suggest that many engineers work on engineering tasks less than 50 percent of the time in any given work week.

A fundamental underpinning of the book articulated in the first part is a constant focus on customer value. The focus has to do with learning what the customer values, developing projects that will deliver value, and executing projects in a manner that maximizes value production and minimizes nonvalue producing activities.

The second part of the book describes 12 tips for making one's projects more efficient, or lean in his parlance. These are not rocket science nor are they things that are likely to be unfamiliar to readers. However, what might be new is the potential they hold for eliminating waste, or conversely, the magnitude of waste one might be incurring if he or she is not using the methods. I think it will be easy for readers to look around their organizations and projects and see opportunities to apply these methods.

Each method is discussed according to a common template (one of the methods is using standard templates) that has the following items:

- A brief description of the problem being solved
- Lean countermeasures that help slash waste
- A step-by-step implementation guidance
- Identification of what one will need and who one will need
- How to measure success

As an example, method no. 6, "Staged-Freeze Specifications" addresses the trade off between speed (cycle time) and risk. Although not directly stated, Mascitelli acknowledges the motivation for a waterfall life cycle as the apparent security of having a firm and complete foundation prior to proceeding with further development. In contrast to this, he suggests breaking the large documents and their corresponding phases into smaller portions that can be produced in smaller amounts of time and that allow some concurrent activity to occur. Indeed, the method description suggests a spiral development approach with its focus on risk reduction. If one implements this method, the proposed measures of success are number of months saved in project execution, reduction in high-impact requirements changes, and customer satisfaction.

The last two parts of the book illustrate the methods and thinking discussed in the first two parts. Regarding the special case of new product development, I particularly enjoyed the example illustrating the power of a product-line approach. In this example, an instrumental lesson concerns delaying customization until as late as possible in the production process. He calls the approach "mass customization."

In summary, this book is written in a straightforward manner. Indeed, the author mentions using his methods to produce the book. It was, after all, a project. The book is filled with illustrations and examples that should resonate with readers; they make clear the problem addressed and the proposed solution (that is, countermeasure). I recommend this book particularly to those who find themselves in organizations where the reasons for certain practices, reports, and activities are no longer known and no one has questioned them recently. This book provides the thinking for asking the reason why and offers suggestions about what to do. On the other hand, those implementing projects with little discipline will find ideas about action that they can take today that will have an immediate payoff.

Understanding Open Source Software Development

Joseph Feller and Brian Fitzgerald. 2002. Great Britain: Addison-Wesley. 220 pages. ISBN 0-201-73496-6. (CSQE Body of Knowledge areas: Software Engineering Processes)

Reviewed by Scott Duncan
softqual@knology.net

The authors state that they wrote this book because open source software (OSS) has proliferated into mainstream commercial organizations and has been adopted by governments "as an alternate path for software development." OSS is no longer an academic exercise and the processes used in its development have found their way into "large traditional software houses like Apple, IBM, Netscape, SGI, and Sun Microsystems." The authors make a point of noting that "software users recognize the quality and stability of the products, the economic advantage of shared cost and shared risk in software ownership, and the technological advantages of building open and modifiable platforms." So it is not merely that OSS products are available "at little or no cost." (Even given this latter point, the authors remind readers that "the major portion of the total cost of software development is