Editor: Warren Keuffel ■ wkeuffel@computer.org

A Software Configuration Management Primer

Philipp K. Janert

Software Configuration Management Patterns *by Stephen P. Berczuk and Brad Appleton, Addison-Wesley, 2002, ISBN 0-201-74117-2, 218 pp., US$39.99.*

In the center of the triangle of project-related tasks, which is formed by low-level coding, high-level design, and overall project management, lies a little-noticed area that nevertheless has great potential impact on a project's success: software project support. Its purpose is the design and maintenance of software developers' work environment, and it comprises tasks such as build and configuration management, tool selection and application, and hardware and workspace setup. Compared to the three tasks mentioned earlier, it's received surprisingly little attention in the professional literature and conscience.

In my experience, newly formed software development teams often spend an inordinate amount of time debating possible choices for the work environment, and still (more often than not) end up making ad hoc decisions. But it's also uncomfortably common for teams to fundamentally misunderstand or misapply their tools, causing—at best—frustration and productivity loss. All this is unnecessary, because most of these issues have been understood for years, and suitable, mature tools are readily available. It's just another strange case of nondiffusion of knowledge!

This is why Stephen P. Berczuk and Brad Appleton's *Software Configuration Management Patterns* is such a welcome contribution. In 20 short and readable chapters, they talk about basic and more advanced best practices in SCM. After introducing the main concepts of central repository, private workspace, and mainline development, they discuss why and when to branch and how to handle the subsequent merges. They devote several chapters to unit, integration, and regression testing as necessary activities to maintain codebase integrity.

Practical discussions

What sets this book apart is the directly applicable advice it offers. The authors clearly speak from experience and never slip into vague or abstract generalities. The discussion of various reasons why a team might consider introducing a branch into the source tree is particularly clear and instructive. Most other texts limit themselves to describing the mere mechanics of branching. However, this book explains situations that justify or even require a branch, such as handling post-release fixes, third-party sources, or long-running but independent development tasks within the same project. Some of their ideas are rather original, such as the prerelease branch concept, designed to help stabilize the codebase before a release without requiring an all-out code freeze.

The authors adopt a *pattern* format, where they introduce each concept as a "solution to a problem in context." I have to admit that they failed to convince me of this approach. Firstly, software configuration management is above all a process and casting the discussion into the terms of static entities (patterns) appears pretty artificial. Furthermore, the purposes of the various software configuration management practices are rather straightforward, and the requirement to find a suitable "problem" so that each activity can then be motivated as the corresponding "solution" leads to a narrative that is rather cumbersome and sometimes strangely backwards.

Several topics are mentioned only in passing, such as the interdependency of SCM tools with the build process and the requirements each can place on the adopted directory structure. Also, the book doesn't address release management as a separate activity, and it only briefly treats the politics of configuration management (the integration manager isn't sometimes called the ramrod for nothing). The authors focus strictly on SCM's technical side; they don't mention concepts such as change boards, audits, and baselines.

The book has two useful appendices: one is a list of online resources on related topics, and the other offers detailed descriptions of 12 of the most commonly used SCM tools available.

Overall, Berczuk and Appleton's *Software Configuration Management Patterns* is probably the most timely, hands-on, useful book on this topic today.

Philipp K. Janert is a software project consultant and maintainer of the beyondcode.org Web site. Contact him at janert@ieee.org.

There Is No Silver Bullet

Shantha Mohan

How to Run Successful Projects III: The Silver Bullet *by Fergus O' Connell, Addison-Wesley, 2001, ISBN 0-201-74806-1, 322 pp., US$34.99.*

Fergus O'Connell chose the title *How to Run Successful Projects III: The Silver Bullet* in response to Fred Brooks's 1987 article "No Silver Bullet." While I appreciate that the title might lure readers to pick it up, those of us who are experienced project managers know there's no silver bullet.

To be fair to the author, he does say in the introduction that he hasn't really made a breakthrough; rather, if you follow his 10-step project management methodology, all of your projects will be successful. Moreover, his writing style makes it easy to read and understand this methodology (not withstanding statements such as "Requirements and design phases are a bit like adultery," which do nothing for the book's objectives). Overall, O'Connell has created a good project management "how-to" book.

How to Run Successful Projects III presents a detailed project management process combined with a key concept called the Probability of Success Indicator—a measure of the project's progress through the 10 steps. One could say the PSI, in an indirect way, is a poor man's use of the "earned value" process. For an experienced project manager, the 10 steps are quite well known in some form or another, but for novices, O'Connell's presentation is a great introduction to the process. The PSI, on the other hand, is a useful new measure of the project's progress. You assign a weight to each step according to what it contributes to the PSI. The problem is that it's difficult to accurately assign the PSI contribution. For example, "score low" or "score high" ratings are quite subjective.

O'Connell organized the book into five parts and six appendices. Part 1, on analyzing and planning projects, covers the first five steps of the process, from visualizing goals to managing expectations. Part 2, on reviewing and implementing the plan, covers steps 6 through 10, from leadership style to keeping people informed. Parts 3 and 4 discuss running multiple projects simultaneously and assessing project plans. And Part 5 looks at activities inherent in any project management undertaking: resolving issues, coping with stress, picking the right people, negotiation, meetings, presentation, delegation, and, last but not least, accelerated analysis and design.

The chapters beyond those that describe the 10 steps are good for a quick review. When you don't have enough time for the activities they discuss and you're under pressure to get going with implementation, you'll find the chapter on accelerated analysis and design quite useful. However, if you really want to successfully accomplish the activities, I suggest reading specific books on the subjects. For example, the chapter on picking the right people doesn't talk about adequately assessing the interviewee's technical competency.

How to Run Successful Projects III is a great introduction to project management. The book reads well because the author has tried to keep the language simple and the contents focused. Projects succeed because you plan them well, staff them with people who are right for the job, identify risks and plan for contingencies, and constantly monitor and re-plan to achieve their original goals. It's hard work, but you reap enormous rewards when you succeed. Fred Brooks is still right—there's no silver bullet.

Shantha Mohan is president of Kaveri, a software management technology consulting company. Contact her at shantha_rm@yahoo.com.

IT Architecture: Patterns for Success

Robert C. Larrabee

Beyond Software Architecture: Creating and Sustaining Winning Solutions *by Luke Hohmann, Addison-Wesley, 2003, ISBN 0-201-77594-8, 314 pp., US$39.99.*

Software architecture is a confusing topic for many people. When I discuss this topic publicly, eyes glaze over and brows furrow. When I talk about project success, however, the audience becomes energized and animated. Obviously, architecture (whatever it might be) is essential to success in information technology projects. Luke Hohmann has a track record of bringing a fresh perspective to old problems, and *Beyond Software Architecture: Creating and Sustaining Winning Solutions* continues that trend. He places software architecture in the appropriate business context. Your enterprise wants to sustain success, for which it must first create a good solution, for which it needs a sound architecture. (See the "Suggested Reading" sidebar for more on software architecture.)

Architectural types

Depending on your perspective, an elephant resembles a rope, a tree trunk, a huge leaf, or a mighty wall.

This book describes IT architectures. There are many types of architectures, perhaps even more than Hohmann acknowledges. He introduces *tarchitecture* (technical architecture, which is what most of us mean when we use the unqualified term) and *marketecture* (market architecture). Although not yet common terminology, these ideas are germane to creating and sustaining winning solutions. Like the elephant surrounded by six blind men ("Parable of the Blind Men and the Elephant," Udana 68-69, Buddhist canon), each of whom describe the elephant as he experiences it, "architecture" carries disparate and often conflicting interpretations. Solutions (the actual elephant) lie near the intersection of these different descriptions.

Architecture as a driver

IT architecture is a prelude to functionality. It can be a prelude to success or to a less favorable outcome. So what is the prelude to architecture? The author suggests that you must appropriately allocate business requirements to architectures if you want to both create and maintain good solutions. In keeping with the current trend toward abstraction, Hohmann provides a pattern for strategic product management (actually, a metapattern—a pattern for a pattern). This too is a recent industry trend. The Software Engineering Institute developed a well-received architectural analysis method (the Architectural Tradeoff Analysis Method) and an architectural reuse methodology (Product Line Practice). Other authors (notably, Barry Boehm and David Parnas) have written extensively and compellingly on architecture's centrality.

Architecture as a unifying principle

The concept of alignment is steadily gaining traction with today's business innovators. All efforts must support a final, clear goal; why else would we expend the time and effort? This book suggests that architecture might contribute to this unifying, top-to-bottom total picture. And it's not just a technology picture; it's an enterprise perspective. In 1997, Hohmann wrote *Journey of the Software Professional: A Sociology of Software Development*. This work expanded on classical IT project management concepts, adding humanistic underpinnings to our intuition. *Beyond Software Architecture* likewise expands the architecture field. It's intended for the technical businessperson, the IT strategist (yes, Appendix B is entitled "A Pattern Language for Strategic Product Management"), management, and forward-thinking technologists. People executing business process reengineering could benefit from this book. Like Hohmann's previous book, this one also defies categorization. Most senior technologists and business people would find it interesting and thought provoking. It's clearly written and carefully edited, and a valuable addition to my bookshelf.

Robert C. Larrabee works for ARINC Engineering LLC. Contact him at larrabeerc@ieee.org.

Software Architecture: Suggested Reading

Many recent works discuss this relatively new field. *The Art of Systems Architecting*, by Mark Maier and Eberhardt Rechtin, addresses architecture in a technical context. *Software Architecture in Practice*, by Len Bass, Paul Clements, and Rick Kazman, is the early definitive work. *Software Architect Bootcamp*, by Raphael Malveau and Thomas Mowbray, addresses the Zen of the discipline. The IEEE recently published its architectural standard 1471 titled *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*. More recently, *Design & Use of Software Architectures* by Jan Bosch tackles software architecture and component development. Other current works address component development "in the small" (CORBA and DCOM, for instance), but most books with "architecture" in the title are pitched at a more technically abstract level. In *Beyond Software Architecture*, Luke Hohmann addresses architecture at the enterprise level of abstraction—in its proper business context. The book also addresses scripts, logic layers, and similar aspects of purely technical detail.