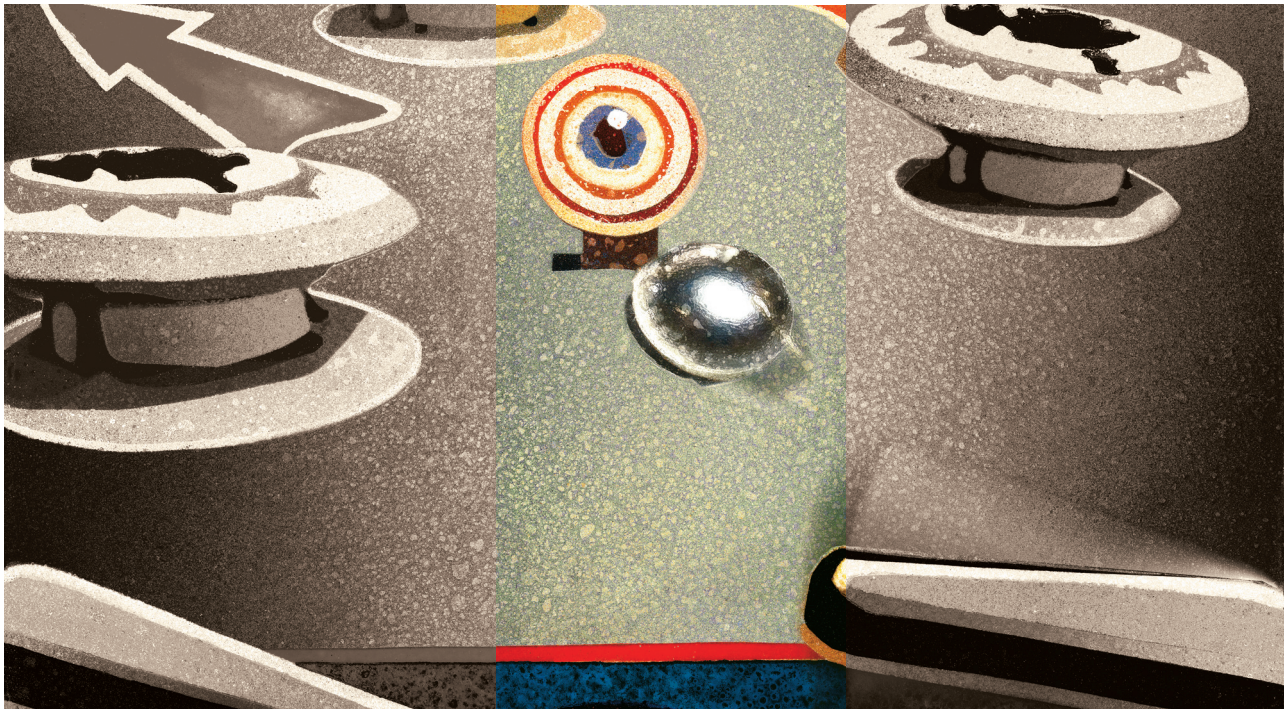

Renewing the Software Project Management Life Cycle

John Favaro, *Informatica E Tecnologia Del Software Spa*

A software project manager from 20 years ago would find much that's unfamiliar in the way projects are managed today. Back then, software development was mostly about engineering, and the spectacular boom-bust-boom cycles demonstrating software's vast potential for influencing business value creation were still years away. We built what we were told to build, managing it the best we could; and after it was over, we breathed a sigh of relief (or despair) and moved on to the next project. Little could we imagine that today we'd bear major responsibility

0740-7459/10/$26.00 © 2010 IEEE

More Information on Project Management

Software project management is renewing itself in unexpected ways, expanding its reach into disciplines that were unrelated or even nonexistent just a few years ago.

The International Institute of Business Analysis (www.theiiba.org) is trying to elevate what we know as requirements engineering to the level of the enterprise, strengthening the focus on business value creation.

Marketocracy (www.marketocracy.com) is an example of the phenomenon popularized in the book Wikinomics (www.wikinomics.com), where talent is discovered and harnessed through social networking technologies. Another example is TopCoder (www.topcoder.com), which "crowdsources" software development solutions from a community of competing talent.

For more information on global software development, see the *IEEE Software* September/October 2006 special issue, with an introduction by guest editors Daniela Damian and Deependra Moitra. James Herbsleb of Carnegie Mellon University has dedicated a course to the subject in recent years, with lectures and readings available on the Web.

IEEE Software Advisory Board member Linda Rising has assembled a unique collection of resources on project retrospectives at www.lindarising.org. Also see *Project Retrospectives: A Handbook for Team Reviews* by Norman Kerth.

Up until recently, the discussion around knowledge as an asset has been relatively myopic, primarily centered around narrow issues such as how to treat intangibles in a company's accounting system. The New Club of Paris (www.the-new-club-of-paris.org) has broadened the discussion, relating it to the discipline of intellectual capital reporting that relates knowledge acquisition to its overall effects on economies and societies.

in deciding what should be built; that we'd have to build it in a borderless, connected environment; and that the criteria for evaluating the success of a software project would be irrevocably altered under the pressure of an unforgiving, globalized economy. Software management practices have been slow to catch up with these changes, but we're beginning to see signs of renewal.

There are many different perspectives on the project management life cycle, but all managers can agree that at its most essential, a project—like a story—has a beginning, middle, and end. Let's examine the ongoing process of project management practice renewal through three articles that offer new perspectives on the Three C's of the project management life cycle: conception, construction, and conclusion.

Conception

Strange though it might seem, the value of a software project might well be determined at birth. In today's grim economic environment, the success or failure of an IT project is often decided at conception by the answer to a simple question: is it innovative? Innovation has always been the lifeblood of IT; in the end, IT is all about finding better ways to do things. The International Institute of Business Analysis (see the sidebar "More Information on Project Management") is encouraging us to replace the term "requirements analyst" and its engineering-related connotation with the term "business analyst," whose job is to help the company squeeze out every bit of innovation possible in order to survive. Google famously encourages innovation by giving its personnel unsupervised time to work on their pet projects. But Google, with its aura of "the best and the brightest," is more like a think tank, and it's natural to wonder whether its approach to innovation applies to the rest of us.

In the Pixar animated film *Ratatouille*, the great chef Gusteau's motto was "anybody can cook." As the film's food critic character observed, great talent can be found in the most unexpected places (in this case, it was a cooking rat). In today's networked society, there are many variations on this theme. The motto of the Marketocracy (see the sidebar) stock-picking site is that "anybody can be a great stock picker." But all such variations on this theme face a crucial problem: it's not enough that there be great talent out there—you have to find it. Marketocracy lets would-be stockpicking gurus build a track record managing a virtual portfolio of stocks, then uses advanced algorithms to identify the most talented.

But is it true that anybody can innovate? And even if it's true, can you find the great innovators around you, even in the most unexpected places? Tony Gorschek, Samuel Fricker, Kenneth Palm, and Steven A. Kunsman, the authors of "A Lightweight Innovation Process for Software-Intensive Product Development," are betting that they can do just that with their—yes, innovative—program Star Search, named for one of the original television shows for talent discovery.

Construction

A series of technical, political, and economic upheavals has made the way in which software systems are constructed nearly unrecognizable from 20 years ago. With the rapid expansion of the Internet in the 1990s, suddenly, such remote places as Australia were literally on the map. New Zealander Rick Mugridge, coauthor of *Fit for Developing Software*, says that "in earlier days, we felt so far away from everywhere. When e-mail technology arrived, it shrunk the world; when the Web arrived, our world shrunk again—being up with the latest play, taking part, being connected."

With the fall of the Berlin Wall and the rise of the so-called "emerging economies," it's no longer exotic to have projects with participants located in India, Uzbekistan, or Estonia. *IEEE Software* has documented much experience with global software development in these very pages and Carnegie Mellon has instituted a course on the subject (see the sidebar).

But it's not enough to collect anecdotal evidence and write about it in magazines and journals. If we're to stay viable as a discipline of software engineering, we have to take that anecdotal evidence and fold it back into our standards. In order to do that, we must first confront the current state of our standards with this new reality. John Stouby Persson and Lars Mathiassen's article "A Process for Managing Risks in Distributed Teams" is an example of this process: in today's multitechnology, multicultural, multiparadigm IT landscape, we've found that global, distributed-system construction brings both benefits and risks. The authors describe the development of their risk plan according to an established standard. The results of experiments like this will help us renew our standards for future efforts.

Conclusion

When a project has concluded, the computers turned off, the source files archived, and the personnel reassigned, how do you measure its success? IT managers have suspected for a long time that a successful software project doesn't just deliver the right system built the right way, but rather makes other kinds of contributions to enriching the enterprise, such as providing personnel with valuable training and experience. Much of the renewed interest in project retrospectives (see the sidebar) is rooted in the realization that success isn't just about building the system, but learning from the experience, and managers haven't done their jobs if they haven't captured those benefits, too.

The idea that a successful software project also generates intangible benefits for the enterprise has been around for a while, but we're seeing renewed interest in the discipline of intellectual capital reporting. The New Club of Paris is bringing intellectual capital reporting to the attention of IT managers and even national governments as a fundamental tool in their strategies for remaining competitive in the knowledge society (see the sidebar).

These new approaches to software project evaluation and measurement illustrate how some of our most sacred tenets are coming under scrutiny—including the very principle of software measurement itself. Last year, Tom de Marco wondered whether his influential management maxim, "you can't control what you can't measure," introduced over three decades ago, needs revision in view of today's value-oriented IT environment (*IEEE Software*, July/August 2009, pp. 96, 95).

The software industry has relied on the Standish Group's Chaos reports for nearly two decades for insight about software project success and failure. The damning evidence they presented led the way to popular awareness of application bloat and unused features (neatly summarized in Chet Hendrickson's paraphrasing of an old television publicity commercial on weight loss that "inside every large software project there is a small software project trying to get out") and was a major impetus in the growth of the agile movement. Yet the Chaos reports have been controversial from the beginning, and J. Laurenz Eveleens and Chris Verhoef take another careful look at the Standish Group's approach to project measurement in "The Rise and Fall of the Chaos Report Figures."

The articles you'll read in this section are an example of how the software project management body of knowledge is gradually being renewed across the entire life cycle. Hopefully they'll stimulate you to make your own contributions to this process of renewal.

About the Author

John Favaro is a senior consultant at Informatica E Tecnologia Del Software Spa (INTECS) in Pisa, Italy. Contact him at john@favaro.net.