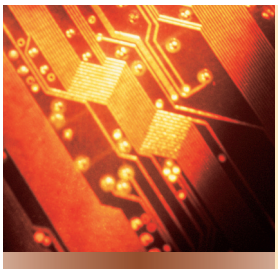
The Agile Methods Fray

Tom DeMarco, Atlantic Systems Guild
Barry Boehm, University of Southern California



Two of software's leading practitioners debate the particulars of implementing agile methods.

In Computer*'s January 2002 issue, Barry Boehm presented a fresh look at a set of software development methods often referred to as agile or extreme programming ("Get Ready for Agile Methods, with Care," Jan. 2002, pp. 64-69). This favorable assessment by one of the software establishment's leading lights prompted the latest of several e-mail dialogues between Boehm and software luminary Tom DeMarco, who strongly advocates that the software establishment begin moving toward agile methods.*

Michael Lutz, Area Editor

FRAMING THE DEBATE

Tom DeMarco: Barry, I was delighted to see you leap into the agile methods fray. Before you came along, the matter seemed to be framed as a debate over the resolution that agile methods are good and should be adopted in place of our current fixed processes.

Those on the pro side—Jim Highsmith and others—argued that the current approaches are broken and should be replaced. Those on the con side—Steven Rakitin, for example—said that agile approaches are just hacking by another name and that we shouldn't abandon our disciplined processes now that we're finally getting them right.

I look at these two camps as the resolution's Trotskyites and czarists: The first camp argues for revolution while the second camp remains determined to retreat "not one millimeter" from the line drawn in the sand at CMM Level 3.621. Your article found a sensible middle ground, identifying some baby to be saved and some bathwater to be replaced.

Barry Boehm: Well, Tom, I've been delighted to capitalize on your neat characterization of the "agile methods fray" in terms of Clausewitz's counterpoint between armor and mobility in military operations. Unfortunately, what we see in both software development and military operations is a tendency for the pendulum to swing back and forth between extremes. Yet in most cases, we need a balance between armor and discipline and between mobility and agility. Actually, though, I would say that the leaders in both the agile and plan-driven camps occupy various places in the responsible middle. It's only the overenthusiastic followers who overinterpret "discipline" and "agility" to unhealthy degrees.

DeMarco: Amen to that. It's fortunate that our field's practitioners are better at finding the sensible middle ground than are the professional advocates.

Since you are our industry's first champion of risk management, I do wish you had presented agile versus traditional process in terms of risk. The agile methods provide a tradeoff between speed and risk. So they are not inherently good or bad, but they trade off between one thing that is good, *speed*, and another that is bad, *risk*. In choosing an agile approach, the manager says, "I will sacrifice some sureness about eventual successful completion to improve my odds of fast successful completion." This tradeoff makes good sense provided that speed really is that important and that everybody understands the increased risk.

Boehm: I tried using risk in the article to help determine "how much planning or agility is enough" by addressing the risks of doing too much or too little of each in given situations. But I definitely agree that your speed versus risk tradeoff is another important dimension to consider. Trading off explicit knowledge captured in documentation for tacit interpersonal knowledge is the main way agile methods achieve more speed while bounding risk.

MAN OR SUPERMAN?

DeMarco: I detected a strange false note in your use of the term "premium people." This phrase sounds so unlike you that I wondered if it might have been your evil twin who wrote those words.

What are premium people, Barry? Are they Nietzche's supermen? Are they the Alphas that Aldous Huxley wrote about in *Brave New World*?

Boehm: I didn't intend much more with that term than what you have observed about the limited supply of software talent—that organizations

low on the talent scale will occupy a more dangerous speed-versus-risk tradeoff curve than organizations high on the talent scale.

DeMarco: I would feel entirely comfortable with what you wrote if you had replaced each occurrence of the term "premium people" with "superbly trained people." The difference is obvious: *premium* implies an innate condition while *superbly trained* implies an acquired—or acquirable—condition.

For example, I gave myself the luxury of sitting in on one of Kent Beck's XP immersion weeks and came away impressed by how much people grew during the experience. He teaches an approach to the key implementation steps—specification, versioning, design partitioning, testing, and so on—as an exercise in skill-building. When they finish that week, the course participants have powerful new capabilities to produce a better design than each could produce alone.

Boehm: Kent's XP training is great. But with XP and more disciplined new methods—such as the Personal and Team Software Processes—reporting early success rates, I think we're still seeing how they work with early adopters. We have yet to determine how these methods work with the late majority and laggards, although good mentoring seems to help both.

In another dimension, I would say that XP training is a necessary but not sufficient condition for success in many applications. Most application projects also need at least one person with both strong software skills and strong application domain skills—Bill Curtis's "keeper of the holy vision"—and the domain skills take a lot longer than a week to acquire.

A discussion I had recently with Alistair Cockburn at our USC-CSE Agile Methods workshop tackled the question, "What do you do if you have only one expert who has both strong software skills and strong application domain skills, but you have five projects that need her skills?" Putting her on one of the projects makes that one agile, but it makes the other four risky. Sharing her across all five projects will require making some knowledge explicit through documentation. I've encountered situations like this fairly frequently.

DeMarco: This example misses the point. Part of our 20-year-long obsession with process is that we have tried to invest at the organizational level instead of the individual level. We've spent big bucks teaching the organization how to build systems. If agile means anything to me, it means investing heavily in individual skill-building rather than organizational rule sets.

Agile means investing heavily in individual skill-building rather than organizational rule sets.

Most of the companies I visit that don't have enough "superbly trained people" today don't have them because they haven't even tried. Once they begin to focus their energies on building individual skill sets, we'll come to see the problem you and Alistair discussed as merely transitory.

Boehm: I agree that the increased focus on individual skill development in both agile methods and personal software processes is long overdue. But I maintain that bravely following some of the agile principles, such as "satisfy the customer, focus on working software, and trust motivated individuals," runs a high risk when the motivated individuals don't have the requisite domain skills. For example, we've had teams that tried to satisfy a customer's desire for a natural-language-processing capability by focusing on toy versions of working NLP software, without realizing they were nowhere close to a robust system. Without some explicit documentation and an external review framework, it's harder to tell when such projects run off course.

PLAN OR SUPERPLAN?

DeMarco: At first, your characterization of the Software Engineering Institute's Capability Maturity Model and its ilk as "plan-driven development" charmed me. But then I started to feel it was all wrong. Extreme programming involves tons more planning than most CMM organizations ever do. There are significant, weighty, and complicated planning steps each day about which versions to tackle next, what should be in each one, what tests will justify the next version and eventually prove it, and how the design should be partitioned. Because developers revisit many of these decisions again and again—that's what refactoring is all about—the mental planning muscles tend to get a lot more exercise in XP than in any fixed process-driven approach.

Looked at in this fashion, the CMM-like processes might better be described as "boilerplate-plan-driven."

Boehm: I agree completely that the software CMM is way too easy to implement with bureaucracy and boilerplate. I've been encouraged to see it being replaced by the Capability Maturity Model Integrated, with its emphasis on risk management and integrated teaming. You can use a risk-driven documentation approach to avoid documenting items that pose low risks if left undocumented, and to avoid documenting items that invite high risks when you do try documenting them—such as GUIs. The same holds true for risk-driven activity levels for peer reviews and independent quality assurance. The rapprochement of CMM and CMMI leaders toward agile methods, exemplified by Mark Paulk's article "Extreme Programming from a CMM Perspective" (*IEEE Software*, Nov.-Dec. 2001, pp. 19-26), is an encouraging trend.

I can still find many applications in which the requirements are relatively stable and a preplanned architecture can successfully accommodate later increments. In such cases, believing "you aren't going to need it," and

revisiting decisions again and again becomes unnecessary rework and an insult to your customer. However, I see definite trends toward more applications with highly dynamic or emergent requirements, which require using agile methods.

DeMarco: After almost 20 years of industry-wide process obsession, the typical process I encounter in client companies has become much too document-centric, resulting in the documentary bloat now endemic in our field. Each new change in process has tended to be additive: Gee, we saw this kind of problem on project *x*, so we'll add a new component to our process and impose it on all projects. Each change has added to the documentary burden. Nothing ever gets subtracted. Agile methods are a kind of backlash against this widely understood flaw in the resultant fixed process. By neglecting to consider documentary bloat, you left out one of the principal ways that agile approaches can help.

Boehm: I agree that this is a big problem and that agile worldviews help combat it. So does CMM/CMMI Level 5, which advocates that we look at where we're overdoing things like documentation, then trim them back. Unfortunately, too many CMM organizations stop at Level 3, which directly results in documentation bloat—a problem the government brings on itself by asking for just Level 3 in many source selections. ■

Tom DeMarco *is a principal of the Atlantic Systems Guild and a fellow of the Cutter Consortium. Contact him at tdemarco@systemsguild.com.*

Barry Boehm *is director of the University of Southern California Center for Software Engineering. Contact him at boehm@sunset.usc.edu.*

Editor: Michael J. Lutz, Rochester Institute of Technology, Department of Computer Science, 102 Lomb Memorial Drive, Rochester NY 14623; mjl@cs.rit.edu