Going to Extremes

Mark Paulk
Software Engineering Institute
Carnegie Mellon University

Kim Caputo has an "Assumptions Worksheet" in her book CMM® Implementation Guide that I consider very insightful in uncovering some of the cultural issues associated with doing software process improvement successfully. She lists seven assumptions and asks whether the assumptions are considered invalid, somewhat valid, or completely valid and reflected in practice. The first assumption is that "Engineering discipline is required to build quality into products of large size and complexity."

For many ASQ (and IEEE) members, the answer may seem obvious. The SEI's mission of advancing the state of the practice in software engineering implies that engineering discipline is a useful and valuable attribute for software professionals to pursue.

There are, however, those who argue that discipline interferes with creativity, and software development is an art, or craft, that is crucially dependent on professionals who are empowered to exercise their innovative ability. "Process" imposes an unacceptable overhead to getting the work done effectively and efficiently. This seems most problematic in the commercial shrink-wrap industry, but it's a concern across the software community.

I agree that creativity, especially good problem solving skills, is crucial to the success of software projects. I also agree that the most important decision that I can make as a manager or team leader is who is going to be on my team - because it is the people who do the intellectually demanding, design-intensive work, and I want competent people who can work together. I also think that we need to apply engineering discipline if we are to work together effectively as a team...

Unfortunately there are extremists on both sides of the creativity vs discipline, innovation vs process debate who are insensitive to the fact that there are legitimate concerns in both camps. When I hear James Bach debating in favor of "good enough software," I agree with many of his statements about things that should be done, while disagreeing with him on the things he thinks are counter-productive (such as the CMM). "Good enough" captures a tradeoff decision that all companies make either implicitly or explicitly. I would argue that the criteria that make a product "good enough" should be explicitly expressed, but the actual business decision resulting from that process is outside the realm of process management - and I may disagree with the result while judging the process to have been well-defined.

Similarly, when Bach talks about "heroes," my immediate reaction is that the last thing the software industry needs is a demand for heroes… yet when I read his characterization of what a hero is, it seems to simply be "competent professionals who have committed to doing their jobs to the best of their ability." In my mind, a hero is someone who throws himself (or herself) on a grenade, dying to save his or her friends and colleagues. I feel no desire to throw myself on a grenade for Bill Gates or Larry Ellison, but I do want to do a competent and professional job in my chosen profession. Am I a hero in my terms? Certainly not! In Bach's terms? Maybe…

It seems to me that the problem that many opponents of the CMM and process management have is rooted more in HOW processes are defined than in the model. Since the CMM emphasizes WHAT rather than HOW, this suggests that we are talking to different issues. There are process improvement initiatives, ostensibly based on the CMM, that embody the "process nazi" approach. If you've been badly burned by such an approach, it is no wonder that you question the value of process rigor. The advice that I give when consulting on such issues is keep it simple, apply your software design principles of information hiding, abstraction, etc., and keep up the dialog with the folks in the trenches doing the work. Keep in mind that the requirements to be a Level 5 organization are contained in the 52 goals for the 18 key process areas, and

® Capability Maturity Model and CMM are registered with the US Patent and Trademark Office.
all the rest of the material in the book is informative. If you have an 800-page process manual, and I have heard of such more than once, you have a prescription for disaster.

There are a lot of good ideas in Extreme Programming and Lean Software Development, to mention just two of the currently popular "methodologies." I do not agree with every practice they advocate, and there are things that I would like to add, based on my experience with CMM-based improvement, but the bottom line is that there are many different methods for working effectively and efficiently. If we act as professionals, keeping an open mind, we can resolve many of the differences that we have based on objective evidence. In many other cases, we will simply agree to disagree on the basis of style and personal preference - and that's not necessarily a bad state to be in. Often the "best practice" will depend on the application domain, business environment, and other contextual factors as far as the HOW is concerned while addressing the same basic concern at the WHAT level.

Does discipline interfere with creativity? I would argue that discipline should empower creativity, in the same sense that the scientific method is based on a rigorous, disciplined approach to learning about the universe. Those who argue that you can be disciplined OR creative fall into the "tyranny of the OR" as Collins and Porras express it in their seminal book *Built to Last*. While researching the successful habits of visionary companies, Collins and Porras found that the truly visionary embraced the "genius of the AND." Visionary companies don't just try to be balanced - they seek to embrace both extremes, e.g., to be both disciplined and creative.

If you are in an Internet startup or a commercial shrink-wrap company and decide to adopt one of today's popular methodologies, such as Extreme Programming, I suspect you can do quite well with respect to process rigor. If you continually explore ways of improving the ways you do your work and keep apprised of sources of "best practices," such as the CMM or McConnell's *Rapid Development* or the Software Program Manager's Network (http://spmn.com/), then I suspect you will do well with respect to software process improvement… without ever doing a formal software process assessment. While I believe the Software CMM is the best framework currently available for software process assessment and improvement, it is not rocket science. If you ever are assessed against a model such as the CMM, and you have used common sense and self-discipline in applying good engineering and management practices, you should have nothing to fear.