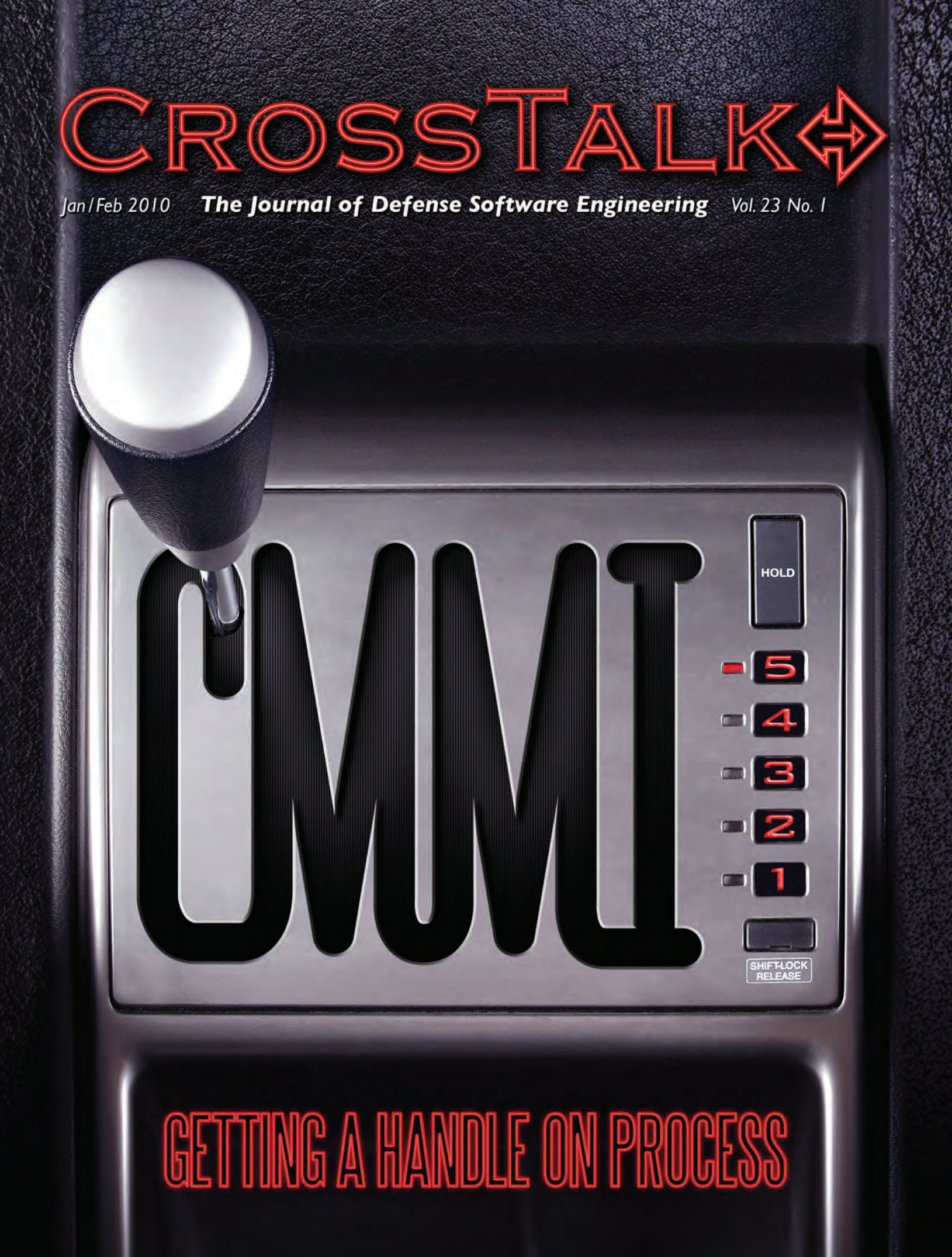
CrossTalk

Jan/Feb 2010 The Journal of Defense Software Engineering Vol. 23 No. 1



GETTING A HANDLE ON PROCESS

Policies, News, and Updates

CMMI: Getting a Handle on Process

Open Forum

On The Cover
Cover Design by Kent Bingham

Departments

CrossTalk

OSD (AT&L)	*Stephen P. Welby*
NAVAIR	*Jeff Schwalb*
309 SMXG	*Karl Rogers*
DHS	*Joe Jarzombek*
Managing Director	*Brent Baxter*
Publisher	*Kasey Thompson*
Managing Editor	*Drew Brown*
Associate Editor	*Chelene Fortier-Lozancich*
Article Coordinator	*Marek Steed*
Phone	(801) 775-5555
E-mail	stsc.customerservice@hill.af.mil
CrossTalk Online	www.stsc.hill.af.mil/crosstalk

CrossTalk, The Journal of Defense Software Engineering is co-sponsored by the Office of the Secretary of Defense (OSD) Acquisition, Technology and Logistics (AT&L); U.S. Navy (USN); U.S. Air Force (USAF); and the U.S. Department of Homeland Security (DHS). OSD (AT&L) co-sponsor: Software Engineering and System Assurance. USN co-sponsor: Naval Air Systems Command. USAF co-sponsor: Ogden-ALC 309 SMXG. DHS co-sponsor: National Cybersecurity Division in the National Protection and Programs Directorate.

The USAF Software Technology Support Center (STSC) is the publisher of CrossTalk, providing both editorial oversight and technical review of the journal. CrossTalk's mission is to encourage the engineering development of software to improve the reliability, sustainability, and responsiveness of our warfighting capability.

S T S C

Subscriptions: Send correspondence concerning subscriptions and changes of address to the following address. You may e-mail us or use the form on p. 34.

517 SMXS/MXDEA
6022 Fir AVE
BLDG 1238
Hill AFB, UT 84056-5820

Article Submissions: We welcome articles of interest to the defense software community. Articles must be approved by the CrossTalk editorial board prior to publication. Please follow the Author Guidelines, available at <www.stsc.hill.af.mil/crosstalk/xtlkguid.pdf>. CrossTalk does not pay for submissions. Published articles remain the property of the authors and may be submitted to other publications. Security agency releases, clearances, and public affairs office approvals are the sole responsibility of the author and their organizations.

Reprints: Permission to reprint or post articles must be requested from the author or the copyright holder and coordinated with CrossTalk.

Trademarks and Endorsements: This Department of Defense (DoD) journal is an authorized publication for members of the DoD. Contents of CrossTalk are not necessarily the official views of, or endorsed by, the U.S. government, the DoD, the co-sponsors, or the STSC. All product names referenced in this issue are trademarks of their companies.

CrossTalk *Online Services:* See <www.stsc.hill.af.mil/crosstalk>, call (801) 777-0857 or e-mail <stsc.webmaster@hill.af.mil>.

Back Issues Available: Please phone or e-mail us to see if back issues are available free of charge.

Is There Progress in Process?

CMMI: Getting a Handle on Process is easier said than done. The constant barrage of variables interacting with the human component makes following a repeatable process model a challenge in any industry—but extremely difficult in software. Building assembly line-type products makes process easier, but the defense software industry doesn't typically make assembly line software. In fact, our jobs are anything but repeatable as we strive to meet the challenges of an ever-changing enemy with its own evolving technological advances, cyberterrorism strategies, and constant countermeasures taken to defeat our latest technology.

CrossTalk would like to thank 309 SMXG for sponsoring this issue.

So why put forth the effort to follow a process model when the products change so rapidly? The reason is simple: There are huge benefits accompanying process. Our software team, the 309th Software Maintenance Group (SMXG) at Hill AFB, has teamed in an enterprise fashion with two other AFB SMXGs—the 76th at Tinker and the 402nd at Robins—committing to process improvement in an effort to further progress down the process road on which we embarked many years ago. We continue because past efforts have significantly reduced defects and increased productivity. Process can do that in a way no other process improvement effort can.

Process should not be misconstrued as a substitute for people, individuality, creativity, and ingenuity—essential elements that are collectively known as agility. These are unique factors that must harmoniously remain inside process, enhancing within the nonrestrictive parameters of a good process model. After all, it is only a model.

I've heard the many arguments made in defense of agility versus process. This issue of CrossTalk will explore the virtues of both, also discussing the transformation and incorporation of process with other concepts, ideas, and process improvement efforts.

We begin with Jeffrey L. Dutton's *An Integrated Framework for Performance Excellence*, which addresses emerging value from CMMI and the associated principles for implementation. Mike Phillips and Sandy Shrum continue the CMMI topic with their article *Process Improvement for All: What to Expect from CMMI Version 1.3*. They detail changes to CMMI for Development, Acquisition, and Services—as well to appraisal and training methods—found in the new version, scheduled to arrive in late 2010.

Anyone battling project control issues will want to read *Scope Management: 12 Steps for ICT Program Recovery*, where Carol Dekkers and Pekka Forselius describe a process designed to control project parameters. Ron Abler and Ted Warren discuss the other side of the process coin by looking at how advancements in organizations occur by focusing on work products—not just processes—in *Stealth CPI: Managing Work Products to Achieve Continuous Process Improvement*. Darrell Corbin's *CMMI, Swiss Cheese, and Pareto* comes to us from an appraiser's point of view, sharing a case study describing creative techniques used to achieve 100 percent CMMI compliance in a virtual environment.

Finally, in this issue's Open Forum section, Hillel Glazer explains the puzzle piece matching relationship he discovered between process and non-process in *Love and Marriage: CMMI and Agile Need Each Other*.

This issue offers something for everyone, regardless of what side of the aisle you sit on in the process debate. Getting a handle on process may not be easy, but it's worth it. We hope that this issue assists you and your organization in building, incorporating, integrating, optimizing, and managing your process efforts.

Karl Rogers
Director, 309th Software Maintenance Group
Co-Sponsor

Announcing CrossTalk's Co-Sponsor Team for 2010

Kasey Thompson
CrossTalk

I would like to once again express sincere thanks to the 2009 CrossTalk *co-sponsors. Simply put,* CrossTalk *would not exist without them and their generous financial support. As Publisher, I receive countless kudos—via e-mail and the phone—expressing appreciation for an article or issue focus that contributed to individual or organizational success. These compliments really belong to the co-sponsors, who spark countless themes and help bring us the best authors in defense software engineering. Likewise, it is my pleasure to introduce* CrossTalk*'s 2010 co-sponsor team and offer profound gratitude for their continued support and commitment to this journal. I know firsthand of their vision, caring, and dedication to their industry and it is manifested through support of* CrossTalk. *Each co-sponsor and their organization will assist our staff by lending us their inexhaustible experience in engineering, systems, security, acquisition, tools, processes, models, infrastructure, people, and (of course) software. Co-sponsor team members are identified in this section with a description of their organization. Please look for their contributions each month in our* From the Sponsor *column, found on page 3. Their organizations will also be highlighted on the back cover of each issue of* CrossTalk.

Stephen P. Welby, Office of the Under Secretary of Defense (AT&L) – Director, Systems Engineering

The Systems Engineering directorate is responsible for all matters relating to DoD major program acquisition systems engineering, software engineering, system assurance, and system of systems engineering. The directorate includes the offices of Major Program Support, Mission Assurance, and System Analysis, with responsibilities in acquisition program support and oversight, systems engineering policy and guidance, human capital, software acquisition management and engineering, system integration, and government-industry collaboration. The directorate's 2010 focus areas include workforce development, early systems engineering, simplifying defense acquisition guidance, and promoting best systems engineering practices to reduce risk. See <www. acq.osd.mil/sse> for more information.

Joan Johnson, Naval Air Systems Command (NAVAIR), Systems Engineering Department – Director, Software Engineering

NAVAIR has three Strategic Priorities through which it produces tangible, external results for the Sailor and the Marine. First are its People that we develop and provide the tools, infrastructure, and processes needed to do their work effectively. Next is Current Readiness that delivers NAVAL aviation units ready for tasking with the right capability, at the right time, and the right cost. Finally is Future Capability in the delivery of new aircraft, weapons, and systems on time and within budget that meets Fleet needs and provides a technological edge over our adversaries. See <www.navair.navy. mil> for more information.

Karl Rogers, 309 Software Maintenance Group (SMXG) Acting Director

The 309th SMXG at the Ogden-Air Logistics Center is a recognized world leader in cradle-to-grave systems support, encompassing hardware engineering, software engineering, systems engineering, data management, consulting, and much more. Their accreditations also include AS 9100 and ISO 9000. See <www.309SMXG.hill.af.mil> for more information.

Joe Jarzombek, Department of Homeland Security (DHS) – Director of Software Assurance (SwA)

The DHS National Cyber Security Division serves as a focal point for SwA, facilitating national public-private efforts to promulgate best practices and methodologies that promote integrity, security, and reliability in software development and acquisition. Collaborative efforts of the SwA community have produced several publicly available online resources. For more information, see the Build Security In Web site <https://buildsecurityin.us-cert.gov> and the SwA Community Resources and Information Clearinghouse <https://buildsecurityin.us-cert.gov/swa>. Both provide coverage of topics relevant to the broader stakeholder community.

Want to Become a Co-Sponsor?

CrossTalk co-sponsors enjoy many benefits such as inclusion of a page-long co-sponsor's note, placement of their organization's logo on the back cover for six issues, placement of the Director's name and organization on each issues' masthead, special sponsorship references in various issues, the ability to provide authors from within their community in regard to their sponsored issue, and online placement on the CrossTalk Web site.

CrossTalk co-sponsors are also invited each year to provide direction for future themes and feedback from the software defense community at large. Co-sponsors are also invited to participate in an annual meeting held during the Systems and Software Technology Conference to discuss emerging needs, trends, difficulties, and opportunities that CrossTalk may address in an effort to best serve its readers.

CrossTalk welcomes queries regarding potential sponsorship throughout the year. For more information about becoming a CrossTalk co-sponsor, please contact Kasey Thompson at (801) 586-1037 or <kasey.thompson@hill.af.mil>.

The 2010 CrossTalk Editorial Board

CrossTalk proudly presents the 2010 CrossTalk Editorial Board. Each article submitted to CrossTalk is reviewed by two technical reviewers from the following list. Their insights improve the readability and usefulness of the articles that we publish. We give a very special thanks to all those participating in our 2010 CrossTalk Editorial Board.

Wayne Abba	Abba Consulting
COL Ken Alford, Ph.D.	Brigham Young University
Bruce Allgood	Electronic Systems Center Engineering Directorate
Greg Anderson	Weber State University
Brent Baxter	Software Technology Support Center
Jim Belford	OO-ALC Engineering Directorate
Gene Bingue	U.S. Navy
Lt. Col. Christopher A. Bohn, Ph.D.	HQ Air Force Special Operations Command
Mark Cain	309th Software Maintenance Group
Dr. Alistair Cockburn	Humans and Technology
Richard Conn	Retired (formerly with Microsoft)
Dr. David A. Cook	Stephen F. Austin State University
Rushby Craig	309th Software Maintenance Group
Les Dupaix	Software Technology Support Center
Monika Fast	309th Software Maintenance Group
Robert W. Ferguson	Software Engineering Institute
Dr. Doretta Gordon	Southwest Research Institute
Dr. John A. "Drew" Hamilton Jr.	Auburn University
Gary Hebert	538th Aircraft Sustainment Group
Tony Henderson	Software Technology Support Center
Lt. Col. Brian Hermann, Ph.D.	Defense Information Systems Agency
Lt. Col. Marcus W. Hervey	Air Force Institute of Technology
Thayne Hill	Software Technology Support Center
George Jackelen, PMP	Software Consultants, Inc.
Deb Jacobs	Focal Point Associates
Dr. Randall Jensen	Software Technology Support Center
Alan C. Jost	Raytheon – Network Centric Systems
Daniel Keth	Software Technology Support Center
Paul Kimmerly	U.S. Marine Corps
Theron Leishman	Northrop Grumman
Glen L. Luke	309th Software Maintenance Group
Gabriel Mata	Software Technology Support Center
Jim McCurley	Software Engineering Institute
Paul McMahon	PEM Systems
Dr. Max H. Miller	Raytheon Integrated Defense Systems
Mark Nielson	Software Technology Support Center
Mike Olsem	U.S. Army
Glenn Palmer	L-3 Communications, Inc.
Doug J. Parsons	Army PEO Simulation, Training and Instrumentation
Tim Perkins	500 CBSS/GBLA (Atmospheric Early Warning System)
Gary A. Petersen	Arrowpoint Solutions, Inc.
Vern Phipps	Arrowpoint Solutions, Inc.
David Putman	309th Software Maintenance Group
Brian Rague	Weber State University
Kevin Richins	Arrowpoint Solutions, Inc.
Gordon Sleve	Robbins Gioia LLC
Larry Smith	Software Technology Support Center
Dr. John Sohl	Weber State University
Elizabeth Starrett	OO-ALC Engineering Directorate
Tracy Stauder	OO-ALC Engineering Directorate
COL John "Buck" Surdu, Ph.D.	Army Research, Development, and Engineering Control
Dr. Will Tracz	Lockheed Martin Integrated Systems and Solutions
Jim Van Buren	Charles Stark Draper Laboratory
David R. Webb	309th Software Maintenance Group
Drew Weidman	Weber State University
Mark Woolsey	Software Technology Support Center
David Zubrow	Software Engineering Institute

An Integrated Framework for Performance Excellence

Jeffrey L. Dutton
Jacobs Technology, Inc.

After 10 years of utilizing and consulting customers in CMMI®, the author uses his experiences to outline three "driving principles" of performance improvement (focus on business issues /performance goals, involved leadership/process ownership, and rapid improvements); compare four improvement approaches (CMMI, Lean, Six Sigma, and the Information Technology Infrastructure Library [ITIL]); and show how an integrated CMMI-based framework has become a highly successful approach for improvement.

My company's journey with CMMI began as a member of the Version 1.0 Product Team. Since supporting that effort in 1999 and 2000, we have consulted with customers who have adopted CMMI, have adopted it ourselves on several occasions, and have become a partner with the SEI for Appraisal Services and Training. We have also found ourselves part of a growing community that continues to discover the emerging power and value of the CMMI model suite. The driving principles that have emerged from this body of work are:

- An improvement effort must be focused on achieving real business goals and helping the organization execute its business strategy.
- If real results are needed in a meaningful time frame, leadership must be involved in a direct and real way, and process doers should own their own processes.
- To be of significant value, improvements must be accomplished at the *speed of business*: often in weeks or months, rarely in years.

We were driven by customer needs to attempt integration of the CMMI framework with other improvement approaches. We have now, along with our customers, integrated Lean Thinking, the ITIL framework, and Six Sigma mechanisms into the CMMI framework. Surprisingly, we found that CMMI models consistently provided a synergistic integrating framework for these other approaches, resulting in CMMI-based performance improvement capabilities that far outreached any one of these approaches alone.

It is worth mentioning that all of our initial efforts were in support of customer and internal software development organizations. The lessons learned in this article are all directly applicable to small-to-large software organizations. Our first integration of Lean Thinking was accomplished by integrating the Lean software development constructs [1] into the CMMI for Development (CMMI-DEV) framework. The surprising results (discussed in this article) encouraged us to approach the integration of other improvement approaches with a more positive and hopeful view. These ideas were initially briefed by the author (in an acquisition context) to the DoD-sponsored Software Acquisition Fall Workshop (2007).

In this article, I will first discuss the three *driving principles* of performance improvement. A brief comparative discussion of the four improvement approaches follows, and I will conclude with a discussion of the integrated CMMI-based framework.

The Driving Principles

Principle #1: Focus on Business Issues and Performance Goals

An improvement effort without focus on performance goals or the resolution of business issues resembles a missile without a guidance system: You're pretty sure it will land, but you're just not quite sure where.

The performance and/or quality goals should be of real business importance to the organization's or project's leadership, and completely in line with its business strategy. Too often, improvements become overly oriented on the process, as though the process were an end-goal. Focus on compliance with a model (like the CMMI) without improving performance and/or work product quality can have a devastating effect on an organization.

All improvement models and approaches (including CMMI, Lean Thinking, Six Sigma, and ITIL) can be implemented badly—that is, for their own sake. There is no *holy grail* of CMMI achievement (including Maturity Level 5) that will guarantee that an organization will perform better unless it consciously sets out to do so from the beginning. Level 3 organizations may not perform any better than Level 1 organizations—and some actually perform worse!

Figure 1 reflects 10 years of anecdotal evidence of the relationship between focus (the independent variable) and cost or value (dependent variables) in various improvement efforts.

The solid line indicates how business value increases dramatically as focus on real business performance goals is increased. The reasons for this improvement in value are many, but perhaps most important is that when the organization is focused on specific performance/quality goals, CMMI practices and informative components can easily be implemented in the context of the organization's business case. Practices can be implemented as is, or alternatives written that respond better to the business case. Informative components can more readily be sifted for implementation, which turns out to be incredibly helpful.

Notice that *very* unfocused efforts (near the zero abscissa value) can and have actually resulted in improvement efforts that have negative value to an organization.

At the same time, cost can be expected to decrease (as indicated by the dashed line) to an optimum least cost. The primary reason for this is that a sharper focus on business context drastically reduces the rework associated with model implementations that are not helpful to achieving the performance/quality goals of the organization. In one of our most recent improvement efforts, costs were reduced to one-fourth of that for previous, similarly scoped efforts.

Principle #2: Involved Leadership and Process Ownership by Process "Doers"

Process improvement literature discusses the role of management as an enabler—a function that provides resources, allows the process group to do its business, and sponsors the process improvement activity. This approach is fine if slow progress, inefficient and ineffective processes, and reluctant buy-in are the goals.

Leadership focused on performance

℠ SCAMPI is a service mark of Carnegie Mellon University.
® CMMI is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

improvement is much more engaged, providing the business goals and strategy and the realization that performance improvement efforts are exactly about making the business succeed. Leadership is directly and proactively involved in the improvement effort, playing an important role on a weekly or even daily basis. Leadership ensures everyone in the organization knows, by example, that performance improvement is part of the job, not separate from it.

Perhaps the most common mistake organizations can make is to assign the development of processes to a *process group* that is not made up of the process doers. Even if the group is extraordinarily good at eliciting the *process requirements* from the actual users of the processes, its products (the organization's processes) will be resisted or even thwarted. The literature is full of cautions, analyses, and workarounds for the push back or resistance from process doers when the process group deploys a new or changed process. There are discussions of what to do with *heroes* when they cause trouble (!), and advice on how to deal with resistance—even advice on how to *mandate compliance*!

Principle #3: Improvements Should Be Made at the Speed of Business

The speed of business today is driven by rapid changes in markets and in technologies. Few organizations have the luxury of two or three years to make meaningful improvements to their performance. Time frames of a month or a year are more realistic and, most importantly, more responsive to the needs of the organization.

As an academic exercise, it is recognized that improvement velocity has both speed and direction. Direction can be thought of as focus, as discussed earlier. Speed, of course, has to do with how fast the improvement effort produces artifacts and changes in performance or work product quality.

Velocity = Speed of improvement in a focused direction.

We studied the factors associated with the institutionalization of a process in an organization, and came to the conclusion that time was not a significant attribute. Of course, there exists a lower bound on the time to implement and institutionalize processes in an organization, but we believe there is no *standard* time (e.g., two years to Maturity Level 2, four years to Maturity Level 3, etc.). If the organization is focused on achieving important performance/quality goals, and on simultaneously interpreting and assuring model compliance, success is more a matter of achieving the organizational goals while working through the details rather than of the passage of time.

For example, in the case where the cost of an improvement effort was reduced to one-fourth that of similar efforts, time was reduced by 77 percent (11 months versus four years).

I now will provide a comparative discussion of the four improvement approaches we have used most. For each approach, I will provide a value proposition, a look at the downsides, and end with a brief discussion of how well the approach integrates with other frameworks for improvement.

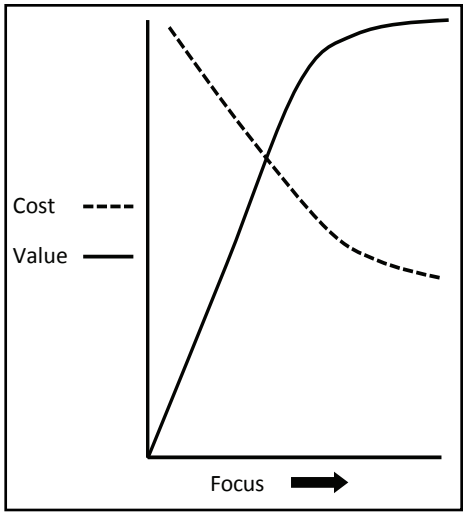


Figure 1: *Dependence of Cost and Value on the Focus of an Improved Effort*

Improvement Approaches

CMMI

CMMI is basically a set of three models—Development, Services, and Acquisition—and SCAMPI℠. The models and appraisal methods were developed by integrated teams composed of people from the SEI, industry, and government (I was an industry member of the CMMI V1.0 product team). CMMI models provide best practices in the three domains of Development (software, hardware, and systems), Services (any kind, including IT), and Acquisition. The models are called CMMI-DEV [2], CMMI for Services (CMMI-SVC) [3], and CMMI for Acquisition.

CMMI's value lies primarily in three areas:

1. The three sets of domain-specific best practices.
2. Practices that enable an improvement infrastructure, allowing process and performance improvements across five different levels.
3. A robust, extensible appraisal method that is recognized for its reliability and credibility.

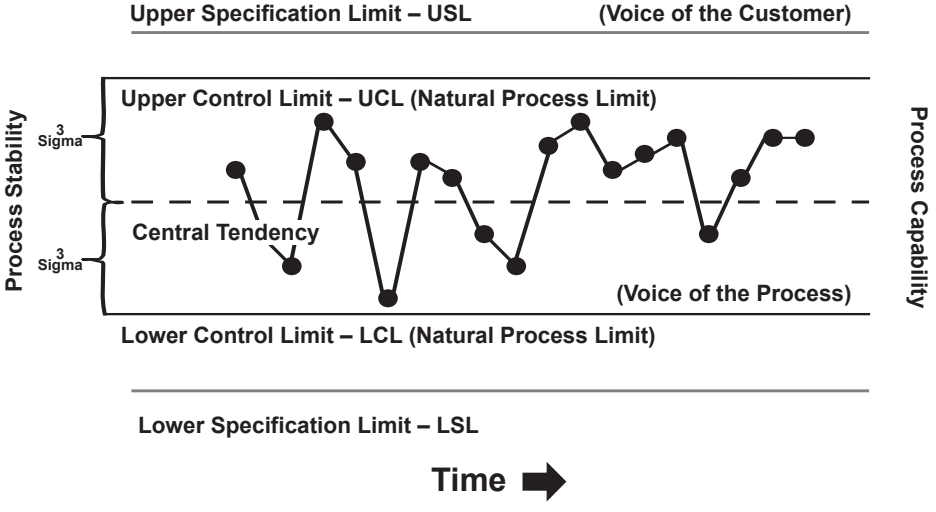
The specific practices contained in the CMMI-DEV model support mature systems, software or hardware development, and have been migrated to enable both Lean and Agile approaches.

The infrastructure practices provide guidance for developing the *organization for improvement*, as well as necessary supporting functions, such as configuration management, quality assurance, and metrics.

SCAMPI allows for responsive, cost-efficient assessments of ongoing improvement efforts, and is extensible to include things such as performance goal evaluation and ITIL. SCAMPI allows for rapid course corrections in improvement efforts as well as rapid and essential learning by the organization, and provides internationally recognized benchmarking.

There are some downsides as well. CMMI does not contain practices or guidance for setting meaningful business or performance objectives, or for formulating improvement strategies to achieve such objectives (with the exception of the Strategic Planning process area within the CMMI-SVC model). It is fair to note, how-

Figure 2: *Sample Process Control Chart*



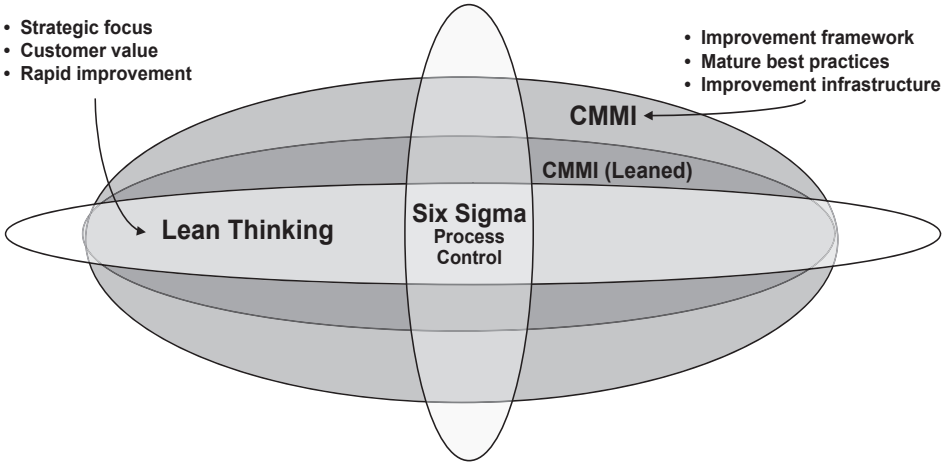


Figure 3: *Overlapping Attributes of Improvement Approaches*

ever, that most models or improvement approaches omit such guidance.

CMMI models are *integrable* with each other in what is called multi-model implementations. Examples are implementations of the CMMI-DEV with CMMI-SVC for engineering development and services organizations. Our experience integrating Lean Thinking, Six Sigma, and ITIL into CMMI-DEV and CMMI-SVC models has revealed that there are no *anti-patterns* or other intractable problems.

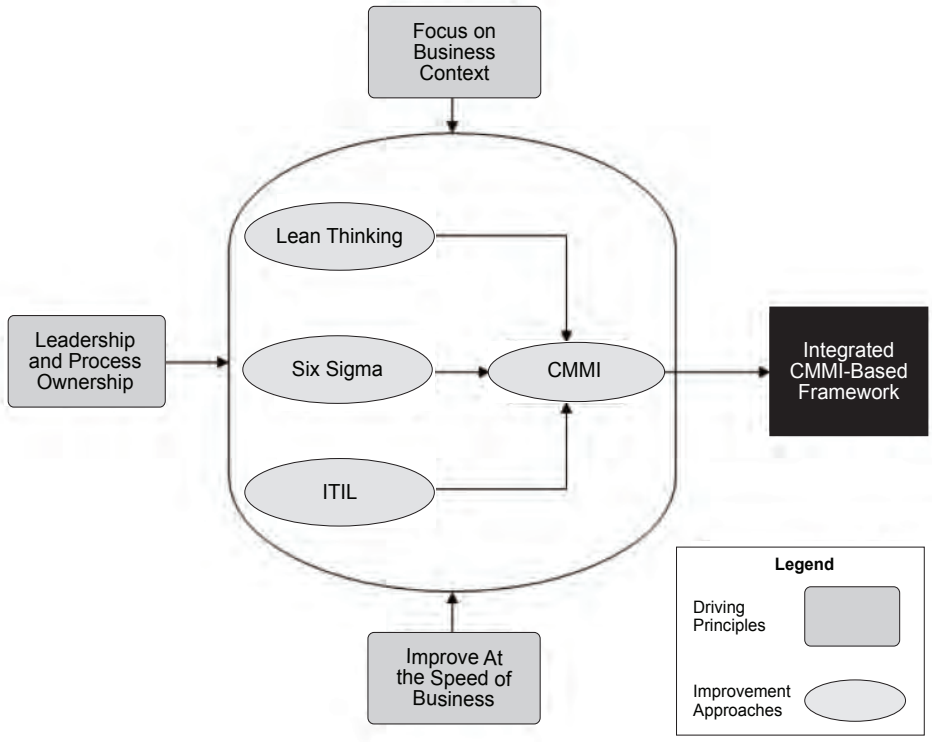
Lean Thinking

Lean is much more than building efficient, streamlined processes and the potential reduction of resources needed to perform a process or service. In fact, these things are secondary to the point of Lean.

Because our customers were beginning to migrate from CMMI to a Lean improvement approach, Richard McCabe (of the Systems and Software Consortium) and I performed an analysis of potential contradictions between CMMI-DEV and Lean software development, and between CMMI-DEV and Agile software development. We rated each specific practice as *Enabling*, *Supportive*, *Acceptable*, or *Unacceptable* in the manner in which they supported Lean and Agile software development. We fully expected to find a few to dozens of *Unacceptables*—but found none [4].

This result, we realized, was attributable to the wisdom of the CMMI framers, who conceived of *Required* (Goals), *Expected* (Practices), and *Informative* (everything else) components. Because the informative components can be adopted according to the business context and case, and because practices can be modified to fit the business case, the CMMI-DEV was found to be very supportive of both Lean and Agile approaches.

Figure 4: *The Four Approaches in an Integrated Framework*



Lean Thinking is about a fanatical focus on delivering value to the customer, waste elimination, setting and attaining performance goals, cadence and synchronization, Agile project management, and fully integrating processes, technologies, and knowledge into a continuously improving framework that responds quickly to customer demands. In Lean organizations, the processes are owned by the doers of those processes, and they are charged by management to make those processes perform better on a continuous, sometimes daily, basis. Lean software development, in particular, is well-defined [1].

On the downside, simultaneous multiple Lean efforts (Kaizen Events) can become uncoordinated and negatively affect one another. Lean, by itself, leaves the definition of an improvement framework or infrastructure to the organization.

Lean has proven fully integrable with the CMMI, ITIL, and Six Sigma efforts. In most cases, Lean is treated as the *lead* approach since it offers speed, focus on customer value, and responsiveness to customer needs.

Six Sigma

Six Sigma is the statistical control and performance prediction capability associated with stable processes. A process that is being statistically managed enables the use of project-level leading indicators rather than the lagging indicators most projects typically use. The common analogy is that managing with lagging indicators is like driving while looking into the rear-view mirror.

A Six Sigma process is simply one for which the specification limits are six standard deviations from the central tendency—and the process is statistically stable. Figure 2 (see page 7) illustrates the basics of a process control chart. The central tendency is normally the mean, but, in certain situations, tracking the mode or median may prove beneficial. The specification limits are set to the desired state or performance, and are considered the *voice of the customer*. The natural process limits are typically set to ± 3 Sigma.

After processes are stabilized, they are made to perform better by either changing the central tendency of the process (typically the mean), and/or reducing the variation due to common causes. Six Sigma tools include regression analysis, tests of hypoth-

esis, process modeling and simulation, process baselining, process control charts, experimental design, and optimization methods.

On the downside, process control is expensive and time-consuming. It depends on process execution data, so longer improvement cycle times (development, service, or acquisition) can be problematic. Organizations wishing to employ Six Sigma process control would do well to ensure that the predicted return on investment warrants the investment.

Six Sigma has been proven to be fully integrable with CMMI model implementations. In fact, it is fair to say that Six Sigma is a preferred approach for implementation of CMMI high maturity (Capability and Maturity Levels 4 and 5). Lean has the advantage of rapid cycle times, which, in addition to its own Lean value, offers the benefit of rapid data collection for Six Sigma studies.

ITIL

Version 3.0 of the ITIL is a knowledge base consisting of a series of five volumes: Service Strategy, Service Design, Service Transition, Service Operation, and Continual Service Improvement. Each volume, in addition to introductory and summary material, is composed of best practices and guidance, including risk analyses.

Certifications associated with ITIL are focused on *individual knowledge development*, from the Foundation to the ITIL Expert and Master levels. For organizations, there is one recognized and one emerging option. Organizations may undergo an audit against ISO 20000, commonly known as the IT Service Management Standard. This standard is somewhat dated, having been founded largely on ITIL 2.0. An emerging and promising option is to adopt CMMI for Services, using ITIL 3.0 best practices and guidance as informative materials in the adoption of the CMMI-SVC model. Several organizations are reportedly targeting on doing SCAMPI Class A benchmark appraisals against CMMI-SVC with integrated ITIL best practices.

On the downside, ITIL 3.0 guidance for the improvement infrastructure is relatively weak. The framework itself does not provide a long-term basis for consistent and continuous performance improvement. The existing *certification* standard is outdated, and does not provide a structure for continuously increasing the level of IT service management.

Several examples of the integration of ITIL 3.0 into an implementation of CMMI-SVC have been a topic of some interest to the CMMI for Services Advisory Group. The marriage appears to be a good one, with CMMI providing practices for the *infrastructure for improvement* and the appraisal method, and ITIL providing IT service best practices. Several SCAMPI Class B and C appraisals have reportedly been done against the CMMI-SVC model, with ITIL best practices. Benchmark (SCAMPI Class A) appraisals are being planned.

Software Defense Application

For more than a decade, the author has used CMMI with private businesses and government defense organizations alike. What comes from these experiences is an integrated CMMI-based framework offering improvements for defense organizations in focus, speed, and cost reductions. The real-world results speak for themselves: a reduction in timelines from Maturity Level 1 to 3 of 25-50 percent; cost reductions from 33-50 percent; and cost avoidance of more than 33 percent for attainment of business performance goals with a corresponding Level 3 process capability.

The CMMI-Based Integrated Framework

As mentioned throughout this article, each CMMI model provides a high-value framework for integration of other improvement approaches. Figure 3 reveals the most salient attributes of each improvement approach, and how these attributes overlap and support one another. For example, Lean Thinking provides a sharp degree of focus on customer value, and provides mechanisms for rapid improvement.

Figure 3 also indicates that CMMI is, practically speaking, collapsed in its application by applying Lean Thinking to its implementation. Six Sigma is basically orthogonal (independent) of both Lean and CMMI, and works well with both approaches. As described previously, ITIL is treated as an extension of the *informative components* in CMMI-SVC.

Earlier, I discussed the three driving principles of performance improvement; Figure 4 illustrates that these principles form the focus of how the improvement approaches are integrated into the CMMI framework. In all cases in which we have enjoyed measurable success, it provides the basic framework for improvement.

As well, Figure 4 depicts the *mind map* of working through the integration of these improvement approaches for a particular environment or business domain, and reminds us of the importance of *first principles*.

This integrated framework for performance excellence has become the de facto approach for improvement in several of our segments, and, we believe, will offer increasing value and responsiveness as we continue learning how to tune this framework. But we can already appreciate that our efforts, and those of many others, have helped to get the most out of CMMI.◆

References

1. Poppendieck, Mary, and Tom Poppendieck. *Lean Software Development – An Agile Toolkit*. Boston: Addison-Wesley Professional, 2003.
2. Chrissis, Mary Beth, Mike Konrad, and Sandy Shrum. *CMMI – Second Edition: Guidelines for Process Integration and Product Improvement*. New York: Addison-Wesley Professional, 2006.
3. CMMI Product Team. *CMMI for Services, Version 1.2*. SEI, Carnegie Mellon University. Technical Report CMU/SEI-2009-TR-001. Feb. 2009 <www.sei.cmu.edu/reports/09tr001.pdf>.
4. Dutton, Jeffrey L., and Richard S. McCabe. *Agile/Lean Development and CMMI*. Proc. of the Software Engineering Process Group Conference Series. Nashville. 9 Mar. 2006 <www.sei.cmu.edu/library/assets/dutton.pdf>.

About the Author

Jeffrey L. Dutton is chief engineer for the IT Support Services segment of Jacobs Technology, Inc., and is a visiting scientist with the SEI. He is certified as a SCAMPI Lead Appraiser, Six Sigma Black Belt, and ScrumMaster. Dutton is a member of the National Defense Industrial Association (NDIA) CMMI Working Group, and serves on both the Steering Committee for the NDIA Systems Engineering Division and the CMMI-SVC Advisory Group. He also is technical chair for the CMMI Technology Conference and User Group.

Jacobs Technology, Inc.
1525 Perimeter PKWY
STE 510
Huntsville, AL 35806
Phone: (256) 971-5527
E-mail: jeff.dutton@jacobs.com

Process Improvement for All: What to Expect from CMMI Version 1.3

Mike Phillips and Sandy Shrum
Software Engineering Institute

The next release of the CMMI Product Suite—an approach that provides organizations with the essential elements of effective processes that ultimately improve their performance—is expected in November 2010. This Version 1.3 (V1.3) release includes improvements to CMMI for Development (CMMI-DEV), CMMI for Acquisition (CMMI-ACQ), and CMMI for Services (CMMI-SVC) models all during the same development cycle. This cycle also includes improvements to the appraisal method (SCAMPI) and CMMI-related training. The improvements planned for CMMI models do not require major changes or retraining for those currently using CMMI.

Since 2000, the CMMI Product Suite—through three separate models—has given organizations a framework for improving their processes (see Figure 1). First was the CMMI-DEV model (created in 2000 and updated in 2002 and 2006), which helps product and service development organizations integrate their software and systems engineering while improving their processes and performance. The CMMI-ACQ model was then released in 2007 to help organizations that outsource, acquire, purchase, or otherwise acquire products and services for their customers. The most recent model, CMMI-SVC, was released in 2009. It helps service organizations to develop quality service processes that enable improved performance, customer satisfaction, and profitability.

During all of this development work on a product line, these models were used by various organizations, and some organizations even used more than one CMMI model. All three models follow the same structure, philosophy, and general approach. Furthermore, there are details that are common across all three models.

Even though these models were released in different years, Version 1.2 (V1.2) was the last release of all three CMMI models. Two major themes drove the changes that comprised V1.2:

1. Refining the CMMI model architecture to create CMMI constellations that served areas of interest (i.e., Development, Acquisition, Services). This change resulted in the creation of the CMMI-ACQ and CMMI-SVC models.
2. Improving the integrity of SCAMPI appraisals that use CMMI models as the reference model to measure process improvement achievement. SCAMPI appraisals are events that follow a standard method for evaluating how well an organization's processes conform to the practices in a CMMI model. When you use a CMMI model and conduct a SCAMPI appraisal, you receive appraisal results that reflect your organization's maturity and capability. Beginner organizations new to CMMI are typically considered low maturity, while those that have achieved exemplary appraisal results are considered high maturity.

Before the idea for a V1.3 release was settled on, the development team created and reviewed model updates that could be released as CMMI Version 1.2a (V1.2a). This version, considered a minor update, was to include updates made only to informative material[1].

The planned model updates for V1.2a were primarily to clarify high maturity practices. These updates were reviewed by a group of CMMI High Maturity Lead Appraisers and the CMMI Steering Group (the executive committee that guides all CMMI work) at a workshop in late September 2008. As a result of the review, the Steering Group determined that making changes to the normative material to modernize the practices for Maturity Levels 4 and 5 was a better choice than only clarifying the practices by updating informative material. So rather than releasing CMMI-DEV V1.2a, the development team is including these and other model updates in the planned release of CMMI V1.3 for all three CMMI models (CMMI-DEV, CMMI-ACQ, and CMMI-SVC).

The Development of V1.3

The CMMI V1.3 project was initiated in January 2009 when the plan to update the CMMI Product Suite was announced. The plan included two months for users to provide final change requests before the development team would begin reviewing and analyzing of the submitted requests.

During March through June of 2009, the development team reviewed more than 1,150 change requests submitted for the three CMMI models and 850 change requests for the SCAMPI appraisal method. Teams were formed to initiate the development of V1.3.

From March until June, the CMMI Steering Group provided criteria to guide the range of acceptable changes to the CMMI Product Suite. The "CMMI Version 1.3 – Plans for the Next Version" [1] was published by the SEI in August 2009. It stated that it will focus on (but not be limited to):

1. High maturity.
2. More effective GPs.
3. Appraisal efficiency.
4. Commonality across the constellations.

It also required that any changes to the CMMI Product Suite (i.e., model(s), training materials, and appraisal method) must meet the following primary criteria, which will likely do the following (also from [1]):

1. Correct identified model, training material, or appraisal method defects or provide enhancements.
2. Incorporate amplifications and clarifications as needed.
3. Accommodate potential additions to model coverage (e.g., safety, security, and life cycle) only by specific direction of the CMMI Steering Group.
4. Decrease overall model size in V1.3 if possible; increases, if any, must not be greater than absolutely necessary.
5. Model and method changes should avoid adversely impacting the legacy investment of adopting companies and organizations.
6. Changes to model architecture will only be incorporated with specific CMMI Steering Group authorization.
7. Changes can only be initiated by Change Requests or by the CMMI Steering Group.
8. Editorial changes to training may be released in advance of V1.3.
9. Changes must not require retraining the nearly 100,000 (as of Dec. 2008) personnel already trained in CMMI. Upgrade training may be needed, especially for instructors, lead appraisers, and appraisal team members.

Each of the recent CMMI releases has

been guided by criteria for acceptable change provided by the CMMI Steering Group. These criteria are typically similar to the criteria in [1]; however, each set of criteria also has some aspects that characterize the version being released.

The correction of defects is an obvious reason for change. The SEI has identified corrections with errata sheets published on their Web site between formal releases. These corrections are then incorporated into the next version. Besides outright corrections, submitters of change requests submit what they think are improvements to the model. These improvements are often clarifications of existing model material. The second criterion encourages clarifications that may be needed to fully understand the intent of model goals and practices.

Excessive model growth is a significant concern, and therefore criteria 3 and 4 seek to limit additions to this release. These criteria couple nicely with criterion 5, which reminds the team to protect the legacy investment of the thousands of organizations who are using the CMMI Product Suite already. Criterion 9 adds a further constraint so that no one will have to start over with the Introduction to CMMI course simply because of the release of V1.3.

The Major Elements of V1.3

Many improvements will be incorporated into the CMMI Product Suite for V1.3. Some of the more significant improvements are described here.

High Maturity Clarifications

As already mentioned, when you conduct a SCAMPI appraisal, you receive appraisal results that reflect your organization's maturity. Beginner organizations that are new to CMMI are typically considered low maturity while those that have achieved exemplary appraisal results are considered high maturity. A focus of current model development is on clarifying the practices associated with high maturity for organizations using the staged approach—and high capability in process areas (PAs) for organizations using the continuous approach[2].

A High Maturity Team was formed. This team's members have been focusing on making changes that improve the clarity of what high maturity is and providing the guidance needed to achieve it. A team leader was chosen from industry project participants to ensure that the improvements made are representative of current best practices in the community.

The High Maturity Team recognized that high maturity practices are currently unclear, leading to a variety of interpretations by users. As they work on V1.3, the team's main objective is to ensure that all CMMI users have a common understanding of high maturity practices in all three models.

Thus far, the team intends to clarify the following:

- The role of informative material in high maturity appraisals.
- The meaning and use of process models and process modeling.
- How business objectives are related to and lead to high maturity.
- What common causes are and how they are expected to be used.
- What high maturity expectations are on individual PA performance.
- The selection, definition, and level of instantiation of subprocesses.

The high maturity changes to the informative material, produced in the V1.2a effort mentioned earlier, are only a part of the full array of change requests now being reviewed by this team in its V1.3 model development effort. Also planned are changes to the structure of high maturity in the model, which includes changes that strengthen the alignment between Maturity Level 4 and 5 practices.

This team's work focuses on the high maturity PAs: Organizational Process Performance, Quantitative Project Management, Causal Analysis and Resolution, and Organizational Innovation and Deployment.

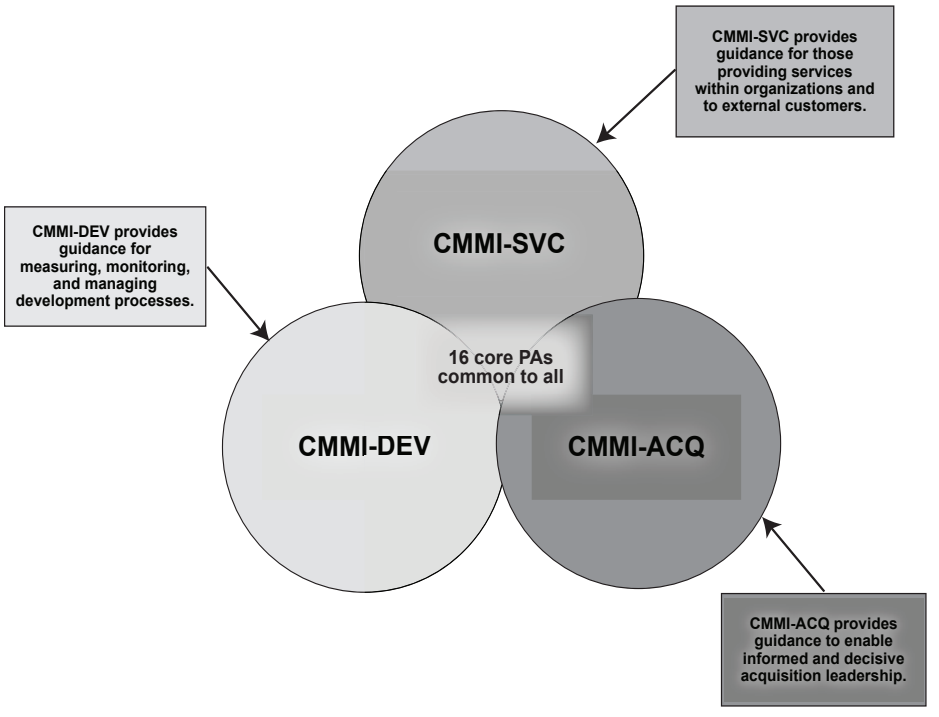
Constellation Commonality

As the development team built on the content of the CMMI-DEV model to create the CMMI-ACQ and CMMI-SVC models, it modernized some of the material in the 16 core PAs that are common to all three constellations. This modernization meant that the development team knew it had to eventually revisit CMMI-DEV and modernize it as well. Even though some differences between the constellations is intentional and makes sense, some of the differences can be eliminated to make the three constellations more consistent and easier to use together.

Here are some examples of the changes being made to improve commonality across the three CMMI models:

- **Core PAs.** These are PAs that appear in all CMMI models. In V1.3, these core PAs can have different expected and informative material. For example, Project Planning can have a specific practice in the Service constellation that is absent in the version of the PA in the Development constellation. Likewise, a few PAs are shared and appear in more than one (but not all) models. Shared PAs also can have different expected and informative material. However, work has been done to ensure that core PAs are as common as it makes sense for them to be. If material can work well in all three models, it is made consistent. If not, the material remains different.
- **Teaming.** There are two different approaches to integrated teaming in CMMI models: in CMMI-DEV, teaming is covered in two goals which are treated as optional, or additions; in

Figure 1: *The Three CMMI Models Now Available*



CMMI-ACQ and CMMI-SVC, teaming is covered in two specific practices in two process areas (Organizational Process Definition and Integrated Project Management). These practices are expected model components and are not optional.

In CMMI V1.3, the development team determined that the best approach to use in all three CMMI constellations is the one used in the CMMI-ACQ and CMMI-SVC models. This work to ensure commonality of the approach to teaming has gained importance since the Team Software Process has demonstrated the performance potential of high performing teams.

- **PA Categories.** There are six PA categories for V1.3: Process Management, Project Management, Support, Engineering, Acquisition, and Service Establishment and Delivery. All PAs that are core must have the same PA category in all three models and this PA category must be one of the following: Process Management, Project Management, or Support. PAs that are not core must be assigned to one of the following PA categories: Engineering, Acquisition, or Service Establishment and Delivery. As a result, Requirement Management will be assigned to the Project Management PA category in all V1.3 CMMI models.
- **Generic goals and practices.** In V1.2 models, generic goals, generic practices (GPs), and GP elaborations are presented differently across models. The CMMI-DEV model presents a portion of these elements in Part One and others are included at the end of each PA in Part Two. In CMMI-ACQ and CMMI-SVC, these elements appear in a single section in Part Two before the PAs. In V1.3, these generic elements will all appear in all three models in one central location as the first section of Part Two.
- **Glossary.** The glossaries in all three models have become inconsistent simply because of the gaps between publication dates of the models. In V1.3 models, the glossaries will be exactly the same, even though some terms defined may not appear in one or more of the models. The format of the glossary will also be modified to differentiate the definition from the notes.

Modernized Practices

Improvements to the practices in multiple process areas will be updated to ensure they are modern and reflect the best practices available.

- **Agile.** Material will be added to the model to help those in Agile environments to correctly interpret practices that may not seem applicable.
- **Architecture-Related Development.** Material will be updated and added to include the consideration of both non-functional and functional requirements during product development.
- **Supplier Agreement Management.** The scope of supplier agreement management will be clarified, particularly in regards to COTS, internal sourcing, and customer property.
- **Organizational Training.** Organizational training practices will be updated so that they apply to more than classroom instruction.

Translations

CMMI models are now available in French, German, Japanese, Spanish, and traditional Chinese. By the time this article is published, a version in Portuguese will also be available. The teams that created these translations have requested that the models' ease of translation be improved. A simple example of a change that can be made to the model to ease its translation is eliminating the use of the word *stovepipe*. This word is one of many that are difficult to interpret into different languages appropriately because its literal meaning is different from how it is used in CMMI models.

Expanded Coverage

A number of change requests have suggested further expansion of CMMI models in new areas. The CMMI Steering Group and the development team do not see this release as being suitable for major expansions like the addition of the two recent constellations, but the team will likely add updated information on architecture, software assurance, Agile, and Lean Six Sigma. The development team has also been encouraged to add more emphasis on customer satisfaction. These types of expansions modernize model coverage without adding new PAs.

Multi-Constellation Coverage

Many students who take either of the one-day supplement courses that cover acquisition or service delivery have commented that many organizations span more than one area of interest. One theme for the V1.3 release is to enable as much sharing of best practices across the constellations as possible. Once some effective pilots are conducted, the development team plans to improve the SCAMPI Method Definition Document to facilitate appraisals that use PAs from multiple constellations.

Appraisal Efficiency

The SCAMPI appraisal method was based (in part) on the CMM-Based Appraisal for Internal Process Improvement (CBA-IPI) assessment method used with the Software CMM, a predecessor of CMMI. The SCAMPI appraisal method moved from the discovery focus of CBA-IPI to a verification focus. This change was designed to save significant appraisal time. The Practice Implementation Indicator Documents (PIIDs) were introduced to reduce on-site appraisal time. These documents list work products that the appraisal team can look for as evidence that a practice was implemented. However, the development team is investigating whether organizations are spending excessive time preparing PIIDs. If they are, the development team will examine ways of upholding appraisal confidence without driving up preparation expenses. The SCAMPI upgrade team is looking for innovative ways to achieve this goal.

Other improvements to the SCAMPI Method Definition Document that will likely be included in V1.3 include:

- Providing SCAMPI support for all three CMMI models by removing problematic terminology, addressing appraisal scoping considerations, and identifying appropriate prerequisites for appraisal team members.
- Correcting all errors identified during the use of SCAMPI V1.2, including common pitfalls encountered by users and problems frequently encountered in reviews of appraisals by the SEI.
- Clarifying the meaning of focus and non-focus projects as well as direct and indirect artifacts.
- Clarifying guidelines for scoping appraisal in a wide range of organization types and sizes.
- Providing guidelines to ensure consistent handling of GPs.
- Resolving issues related to characterization rules and rating rules.

Model Sizing

To meet the fourth criterion that limits the overall size of CMMI models, the development team looks for ways to balance model additions with deletions. Feedback resulting from an effort collecting input from multiple lead appraisers called ATLAS[3]—short for "Ask The Lead AppraiserS," facilitated by Pat O'Toole—was received. This group submitted change requests that identified lower value practices that might be removed to add others now viewed as more important. As mentioned earlier, additional PAs are not encouraged for V1.3.

Upgrade to V1.3

The CMMI Steering Group has approved a significant period of overlap between the release of CMMI V1.3 and the retirement of CMMI V1.2. The development team is also investigating innovative ways of providing information about CMMI improvements to users in draft form. However, no one is encouraged to delay their process improvement programs just to wait for the release of V1.3.

V1.3 Training

Training will be developed that will provide an easy upgrade from V1.2 to V1.3 for all three models. This training will be made available online. The Introduction to CMMI course will be updated as will the three-day Introduction to CMMI-SVC. The current supplement courses for CMMI-ACQ and CMMI-SVC will be retained and updated appropriately.

Development Schedule for V1.3

Given the number of change requests and the span of contentious issues to be resolved by the teams, the development team has been concerned about declaring a schedule for V1.3. The current estimate is to release the three constellations by November 1, 2010, but this date assumes few changes are needed after analyzing feedback on the drafts. Figure 2 provides a high-level view of the V1.3 schedule for the models, and Figure 3 provides a similar view of the schedule for the SCAMPI improvements.

Figure 2 shows that the model development project started in January 2009 and will conclude in November 2010. Preparation activities included planning, forming teams, and defining processes and occurred from January to May 2009. Following that, from June to October 2009, change packages (CPs) were created, reviewed, and approved. CPs are descriptions of planned change based on change requests received; they are reviewed and approved by the development team and then by the CMMI Configuration Control Board (CCB), which is responsible for controlling change to the CMMI Product Suite.

Once these are approved, actual changes to model components are proposed in redlines. These redlines are scheduled to be created, reviewed, and approved from August 2009 to April 2010. These redlines are also reviewed and approved by the development team and CCB.

Piloting is also scheduled to enable organizations willing to pilot improvements before release and provide feedback on their utility. There are three pilot drafts currently planned: a November 2009 draft included changes that improve the consistency of all three models with one another, a January 2010 draft will include many of the model improvements (in particular all of the high maturity improvements), and a June 2010 draft will include all model changes minus the polishing provided by quality assurance (QA). Updates based on feedback from piloting will be made from May to July of 2010. Finally, QA will begin in July to prepare the models for November 2010 release.

By the time this article is published, much of the development of model improvements should be determined and piloting will have begun. If your organization is interested in participating in piloting by reporting on your use of draft versions of a CMMI V1.3 draft model (CMMI-DEV, CMMI-ACQ, CMMI-SVC), contact SEI customer relations at <customer-relations@sei.cmu.edu>. The development team will send you details of how you can receive drafts and how to provide structured feedback.

Figure 3 shows the CMMI V1.3 SCAMPI Method Definition Document development project schedule, which parallels the previously described model development effort.

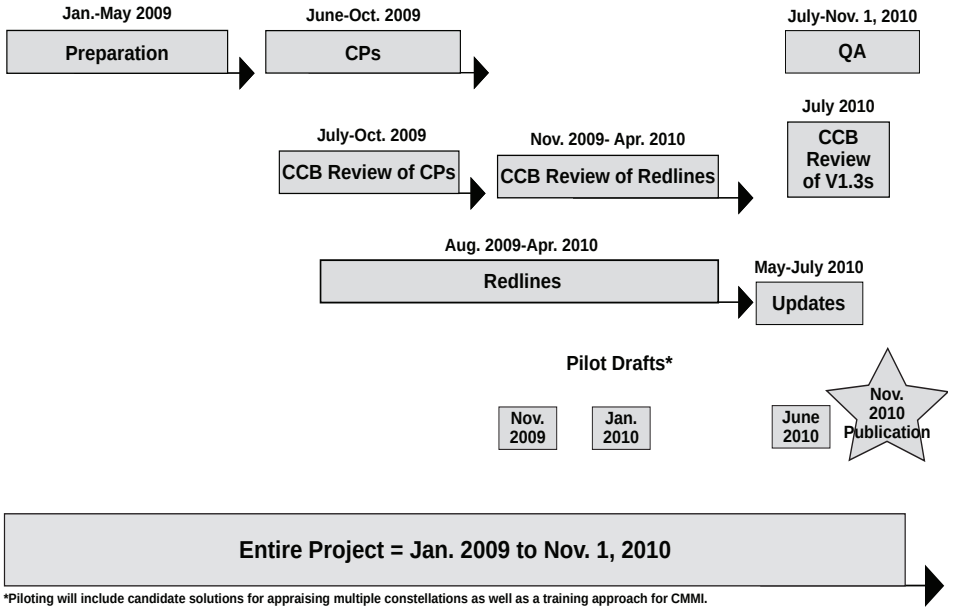
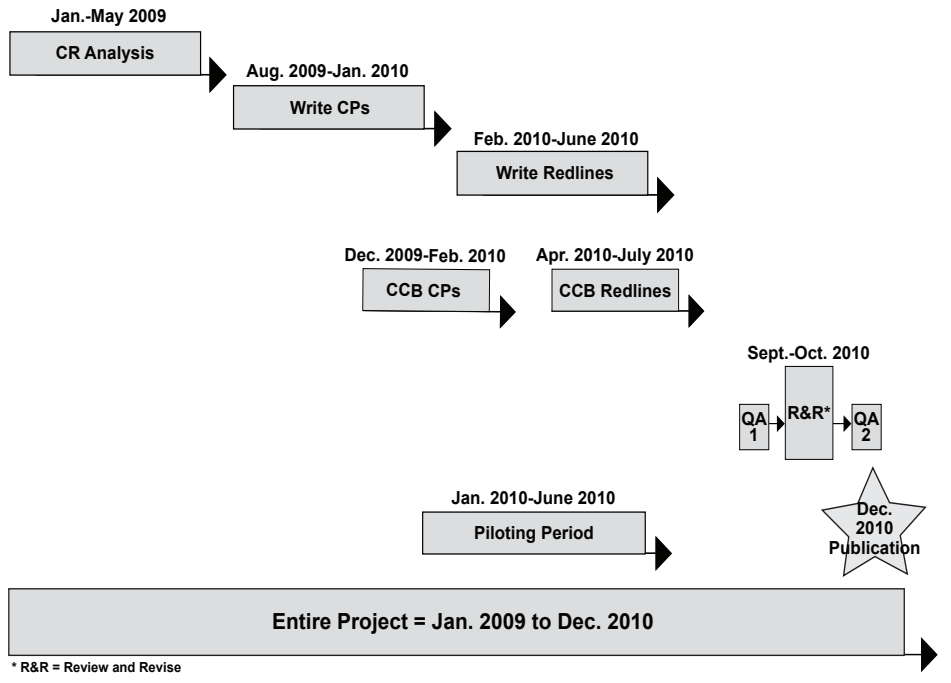


Figure 2: *CMMI V1.3 Model Schedule*

Figure 3: *CMMI V 1.3 SCAMPI Upgrade Schedule*



Software Defense Application

Since many DoD and defense contractor organizations within the software community are currently utilizing CMMI Product Suite V1.2, this article is an extremely valuable primer for the approaching upgrade to V1.3. The value of this article is to alert the software defense community to the specific improvements to the three models (CMMI-DEV, CMMI-ACQ, and CMMI-SVC), the SCAMPI appraisal, and CMMI training methods. Time and money may be saved by knowing what's coming, and perhaps by participating in the SEI's pilot program of the draft models (discussed in the Development Schedule for V1.3 section).

Summary

The improvements included in CMMI V1.3 include high maturity improvements and clarifications, improved appraisal efficiency, and models that have consistent architectures and shared content. At this point in the development life cycle, the improvements have not required major changes to the existing V1.2 product suite. Therefore, CMMI users should continue with their process improvement programs without regard to the release date for V1.3.◆

Reference

1. Phillips, Mike. "CMMI Version 1.3—Plans for the Next Version." *News at SEI*. 7 Aug. 2009 <www.sei.cmu.edu/library/abstracts/news-at-sei/cmmiinfocus200904.cfm>.

Notes

1. The components that comprise CMMI models are grouped into three categories: required, expected, and informative. Unlike the required and expected model components, the informative model material is not considered normative.
2. These two approaches are variations in how appraisals are conducted. However, the idea of high maturity is essentially the same in both.
3. Learn more about ATLAS at <www.pactcmmi.com/pages/atlas>.

About the Authors

Mike Phillips is the program manager for CMMI at the SEI, a position created to lead the CMMI Product Suite evolution. He has authored technical reports, technical notes, CMMI columns, and various articles in addition to presenting CMMI material at conferences around the world. Prior to his retirement as a colonel from the Air Force, he was the program manager of the $36 billion development program for the B-2 stealth bomber at Wright-Patterson AFB. His bachelor's degree in astronautical engineering is from the Air Force Academy, and he holds four master's degrees: nuclear engineering (from Georgia Tech), systems management (from the University of Southern California), international affairs (from Salve Regina College), and national security and strategic studies (from the Naval War College).

SEI
4500 Fifth AVE
Pittsburgh, PA 15213-2612
Phone: (412) 268-5884
Fax: (412) 268-5758
E-mail: dmp@sei.cmu.edu

Sandy Shrum is a senior writer at the SEI. She has been a member of the CMMI development team since its inception in 1998 and has coauthored three CMMI books. Shrum co-published "CMMI-ACQ: Guidelines for Improving the Acquisition of Products and Services" in 2009. Her experience as a writer in the software industry dates back to 1988, when she earned her master's degree in professional writing from Carnegie Mellon University, and her bachelor's degree in business administration from Gannon University.

SEI
4500 Fifth AVE
Pittsburgh, PA 15213-2612
Phone: (412) 268-6503
E-mail: sshrum@sei.cmu.edu

Coming in the March/April Issue

Systems Assurance

User confidence in systems safety, reliability, availability, and maintainability is affected by various overarching factors such as standards, procedures, regulations, and specific verification and certification criterion. And then there's the primary confidence factor: provision of mission-driven functionality as requested by stakeholders.

This edition of CrossTalk will assist readers with knowledge, experience, and proven successes in the Systems Assurance field.

Including the article, "Systems Assurance as a Team Sport" by Robert A. Martin.

Look for it in your mailbox early March!
Issue sponsored by:

Be a CrossTalk Backer

CrossTalk would like to thank the accompanying organizations, designated as CrossTalk Backers, that help make this issue possible.

CrossTalk Backers are government organizations that provide support to forward the mission of CrossTalk. Co-Sponsors and Backers are our lifeblood.

Backer benefits include:

- An invaluable opportunity to share information from your organization's perspective with the software defense industry.
- Dedicated space in each issue.
- Advertisements ranging from a full to a quarter page.
- Web recognition and a link to your organization's page via CrossTalk's Web site.

Please contact Kasey Thompson at (801) 586-1037 to find out more about becoming a CrossTalk Backer.

CrossTalk would like to thank our current Backers:

309th Software Maintenance Group

Cost Analysis Group

OO-ALC Engineering Directorate

309th Electronics Maintenance Group

Scope Management: 12 Steps for ICT Program Recovery©

Carol Dekkers
Quality Plus Technologies, Inc.

Pekka Forselius
4SUM Partners, Inc.

The information and communications technology (ICT) world is "addicted" to dysfunctional behavior and the problem is spreading globally. The sad truth is that the parties in the ICT relationship (the customer and the supplier) are largely co-dependent on a pattern of dysfunction characterized by ineffective communication, fixed price contracts with changing requirements, and eroding trust. This article focuses specifically on the northernSCOPE™ 12-step process for ICT program recovery.

The ICT world's dependence on dysfunctional behavior—specifically ineffective communication, fixed price contracts with changing requirements, and eroding trust—is devastating. Billions of dollars are being spent on rework and enhancements to (defective) software [1]. The latest CHAOS report says that a staggering two-thirds of ICT projects are deemed failures [2], with schedule extensions, cost overruns, and poor quality software more the norm than the exception to the rule. One can find daily news stories where ICT projects are reportedly over budget by several hundred percent, overdue by years, or cancelled after millions of dollars have already been spent.

Almost a decade ago, software suppliers recognized their role in this problem and started investing in process improvement. Similarly, customers focused on improving their technical knowledge in the hope that they could better direct suppliers to implement the right solution. Professional project management and software process improvement initiatives helped streamline the supplier side of ICT programs and projects. However, the issues at play are systemic and involve customers and suppliers.

Questions of how to improve this state of affairs—in an industry of advanced technology, bright project managers, and leading-edge maturity models—led both Australia and Finland to individually investigate further, with results worthy of attention. Their formalized scope management approaches—southernSCOPE [3] and northernSCOPE [4]—have, within the first few years, reversed the trend of failed projects, posted increased ICT program success, and improved customer/supplier relationships. Both approaches are based on project management best practices combined with customer-centric scope management, and illustrate the important role of scope management in building more resilient software that can stand the test of time[1].

> *"Scope management is central to software development and must be integrated especially in regards to time, cost, quality, and risk management."*

Both initiatives examine and advance ICT programs through steps that initialize, scope, split into manageable sub-projects (as necessary), quantify size, cost (on the basis of currency per unit size), and manage and deliver through professional ICT scope management. The results of both approaches are profound: Success rates on ICT projects have skyrocketed and cost overruns have plummeted to levels unprecedented in the ICT industry. In 2005, the International Software Benchmarking Standards Group (ISBSG) proclaimed:

> ... the cost overruns for projects using the southernSCOPE method were found to be less than 10 percent whereas the industry norm was 84 percent. [9]

This article focuses on the concepts of the more recent northernSCOPE, and the differences between it and southernSCOPE will be noted as appropriate. Additionally, the new job role of a certified Scope Manager (CSM), as established by the European Certification & Qualification Association, is discussed. Scope management is not rocket science; however, managing scope is not a natural byproduct of project management.

Why Scope Management?

Introduced in "A Guide to the Project Management Body of Knowledge" (PMBOK Guide) as a knowledge area, scope management can be more important to project success than any of the other individual knowledge areas. As a case in point, 60–99 percent of all defects latent in production software can be attributed to the requirements phase [10]. While project scope management will not guarantee perfect requirements, the simple act of identifying scope delineates what is within the requirements and what is not.

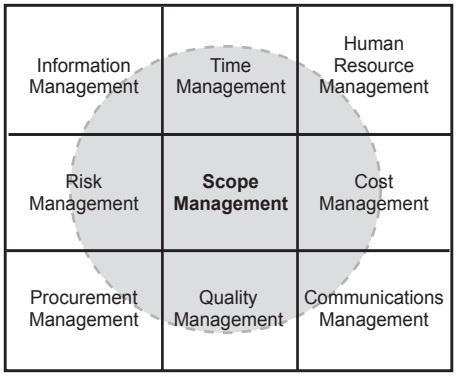
Scope management effectively addresses five out of six of the most common reasons cited for ICT project cost overruns and uncontrolled project growth [8]:

1. Lack of user input.
2. Incomplete requirements.
3. Changing requirements.
4. Technology incompetence.
5. Unrealistic expectations.

Scope management is critical to successful ICT project completion. northernSCOPE places scope management dead center in the overall PMBOK Guide knowledge areas because it involves and interfaces with all eight areas (as depicted in Figure 1).

The PMBOK Guide definition of project scope management is "... the

Figure 1: *Scope Management of northernSCOPE and the PMBOK Guide Knowledge Areas* [11]



processes required to ensure that the project includes all the work required, and only the work required, to complete the project successfully" [12][2]. Scope management is central to software development and must be integrated especially in regards to time, cost, quality, and risk management. There are no scope changes without possible consequences to schedule, budget, and quality, or increases to the risk level of the project. This is true vice-versa as well: If the schedule or budget must be tightened, it requires changing the scope or quality requirements. Otherwise, one increases the overall risk level of the project, including a reduction in quality of project outcomes [11]. The core process of a software development project is *developing the software*. It is not a management process, but an essential object process to be managed (as depicted in Figure 2).

Scope management is best carried out by an independent and knowledgeable scope manager trained in ICT project management, customer relations (communication), software estimation, requirements elicitation, functional size measurement (FSM), change management, and best practices. As a third party usually hired by the customer, the scope manager is an advocate to both the customer(s) and the supplier(s). The role is similar to a construction inspector/coordinator who provides project oversight, governance, measurement, communication, change management, progress reporting, and experience data collection. The European Certification & Qualification Association formalized the northernSCOPE-based CSM job role in October 2007.

northernSCOPE

The northernSCOPE 12-step approach to professional scope management was developed in the late '90s by the Finnish Software Measurement Association (FiSMA) [4]; several CSMs have been trained in how to utilize the approach, with 4SUM Partners and Quality Plus Technologies currently leading the way in this training [13].

The 12 steps are summarized in Table 1. While there may be multiple customers and/or suppliers (e.g., hardware, software, integration suppliers), the scope manager works with all of those affected.

Step 1: Scope Manager Retained, Customer-Driven High-Level Requirements

The northernSCOPE approach is initiated when the customer or software acquirer (or the software supplier) recognizes the need for, and retains, a scope manager for a new ICT program[3]. We advocate using a CSM to ensure that there is a basic level of knowledge and experience with northernSCOPE.

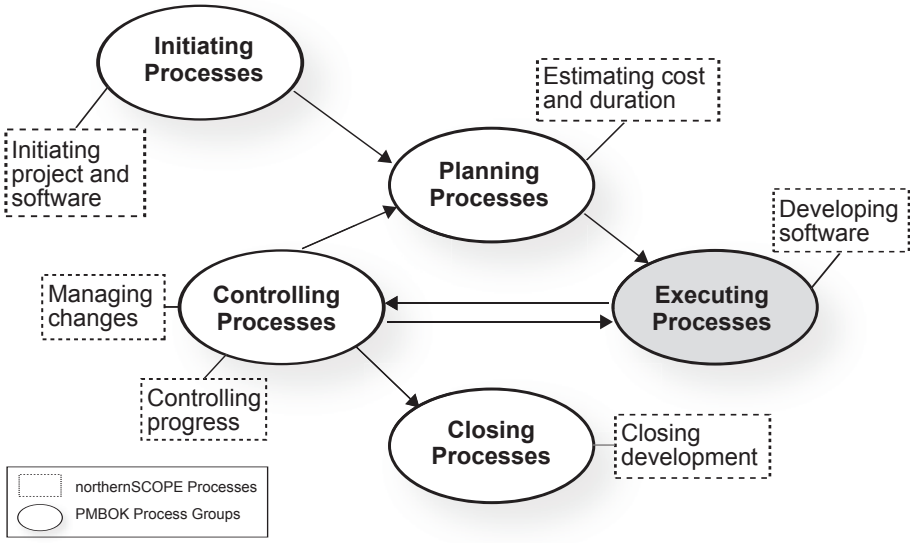


Figure 2: *Mapping northernSCOPE Processes against PMBOK Guide Process Groups*

The first task is a meeting with the customer (typically the program or project steering committee) to outline the roles and responsibilities of the scope manager, the customer(s), and the software supplier(s). This meeting allows the customer to ask questions and to clarify their role. The scope manager also reviews the high-level customer requirements for completeness and clarity, and discusses their envisaged scope and expectations with the customer.

Step 2: Divide Program Into Subprojects

Using the high-level requirements from step 1 and the program subdivision rules (see Figure 3, next page), the scope manager divides the program of work into appropriate subprojects. Note that this is similar to subdividing a construction project into distinct subprojects, each of which is typically managed separately and involves unique tasks. This is important to do as early as possible, preferably before beginning one or more software development projects [14].

Figure 3 illustrates the ICT program subdivision rules, while Table 2 (see next page) depicts seven possible (sub)project types. From the number of rules in Figure 3 and the possible combinations, this approach leads to a larger number of small projects. As with any approach, there are a number of pros and cons. One of the biggest pros is improved manageability so important to program/ project success. Neither the customer nor the supplier should resist this process because it divides an amorphous *big bag of work* into identifiable, manageable, and traceable parcels of work that are easier to mutually discuss and scope than is one large monolithic chunk. While not always

Table 1: *The 12 Steps of northernSCOPE*

Step	Concept	Participants
1	Scope manager retained, customer-driven high-level requirements.	Customer + Scope Manager
2	Divide program into subprojects.	Scope Manager + Customer
3	Scope manager does early FPs for each subproject and estimates total size.	Scope Manager
4	Scope manager and customer determine and analyze quality requirements.	Scope Manager + Customer
5	Customer issues request for proposal.	Customer + Scope Manager input
6	Customer selects supplier based on submitted unit cost per FP.	Customer + Scope Manager input + Supplier
7	Requirements specification developed.	Supplier + Customer
8	Scope manager baselines software size (in FPs) and product development.	Scope Manager
9	Scope manager sizes project changes and cost impact is evaluated.	Scope Manager + Supplier input
10	Scope manager quantifies progress.	Scope Manager
11	Project finishes and customer pays supplier based on FP delivered.	Customer + Supplier
12	Experience data collected and stored.	Scope Manager

1. If the program consists of ICT development and other development work, such as manual process development, re-organizing staff, or technical development, different types of work should be assigned to separate projects.
2. If you apply an incremental or iterative development approach, every increment or iteration should be assigned to separate projects.
3. Different types of ICT development work should be assigned to separate projects.
4. If the program must be stopped consciously for a long time (i.e., to wait for external decisions), the work before and after the break should be assigned to separate projects.
5. If two parts of either product or service development are of a similar ICT project type but differ from each other in the following ways, they should be assigned to separate projects:
   - Development technology
   - Development environment
   - Development team experience
   - Quality requirements of target result
   - Stakeholder dependencies
   - Risk level

Figure 3: *Guidance on Dividing ICT Programs Into Projects and Subprojects* [15]

the case, business projects where software is developed are typically more correctly referred to as programs [12].

Consider an analogy: A buyer of goods requests a fixed price to purchase a series of items—as many as can fit into a single standard-size shopping bag. Without specifically knowing details of the objects they wish to purchase, the constraints are the price (as many objects as possible that can fit), and that the objects must be able to be combined to provide an easy-to-prepare dinner for several people. As such, by discussion and division of what is needed (e.g., cooking utensils, ingredients, food preferences, etc.), both the buyer and the supplier can begin to better scope out the work in advance. Of course, when the supplier(s) are chosen for the work, then the job of planning each piece can be done (the requirements phase in software development) and the contents of the shopping bag are agreed upon.

This step is a successful component of on-time and on-cost delivery in northernSCOPE[4]. The goal here is to create more manageable and definable parcels of work that can be more readily digested and understood by the business customer; however, it is not the intent to create miniscule projects either.

Table 2 lists seven distinct ICT project types. While there are many additional variations on these seven project types, these were the most common in practice based on the original work done with telecom and software industry companies in Finland. See [15] for further details.

The software development life cycle phases—such as requirements through to implementation—are not considered as being independent ICT project types, but rather as phases within each subproject itself.

Table 2: *ICT Project Types* [15]

ICT (sub)Project Type Classification	Description
1. Customer-specific new development project	Creates a completely new customer-specific piece of software.
2. Software product new development project	Creates a new software product. A software product is always developed to be used by more than one customer. A software product may either be standalone-packaged software or an embedded part of another product.
3. Software version enhancement project	Creates a new version of existing software. The existing software may be either customer-specific software or a software product.
4. ICT service development project	Creates a contract-based continuous or temporary ICT service. The service may be, for example, either software or hardware related, and consists of maintenance, support, help desk, or operating service.
5. Package software configuration project	The result of this project is an installed, parameterized, and user-configured software package.
6. Data conversion project	A project where data is moved from persistent data storage of one information system to persistent data storage of another information system. The software developed in a data conversion project is often *throw away* in that it is only used once. Even so, the pieces of conversion software may reside on one or more hardware platforms.
7. Software integration development project	Creates software that provides interface services between two or more information systems.

Step 3: Scope Manager Does Early Function Points (FPs) for Each Subproject and Estimates Total Size

Using the high-level customer requirements for each subproject from step 2, the scope manager performs an initial function-size estimate (where appropriate). Note that the function size measures only the functionality of the software; equally important are the quality and technical requirements (as outlined in step 4). The function size of the program software is the sum of all of its subprojects. Currently, there are five different ISO/IEC standard methods for FSM. The most commonly used method within the northernSCOPE concept is FiSMA 1.1 (ISO/IEC 29881), but all the other FSM methods can be equally applied. However, it is important to specify the used method when publishing the measurement results as the same method should be applied throughout the program. All five FSM standards are detailed in [16].

Depending on the completeness of the functional user requirements (remember that this is prior to a supplier engagement), this estimated function size gives the scope manager and the customer a *ballpark* idea of how big the ICT program could be. If the functional requirements are too vague, then the customer must work to at least identify the business processes to be supported by the software. If a customer cannot state their needs, then a supplier cannot provide an appropriate solution—especially before a request for proposal (to engage a supplier to do work) is issued. Even when a supplier is contracted to perform the requirements elicitation work through to software development, a customer may not be able to fully articulate their needs without the assistance of such suppliers. However, it is always up to the customer to make the final decision about what they need and what they are willing to pay to satisfy those needs.

Several subproject types (and also work such as support and fixes) are inappropriate for sizing with FSM. These include technical upgrades, maintenance work, etc. FPs are a square foot type of measurement used to quantify the functional user requirements for software; without them, the work effort must be estimated using another method, such as an hourly rate or historically based estimate. It is important to communicate to the customer about which subprojects can and cannot be sized using FSM.

Step 4: Scope Manager and Customer Determine and Analyze Quality Requirements

This step is unique to northernSCOPE and examines the quality requirements for

each subproject based on the ISO/IEC 9126 quality model. It is well-established that the non-functional requirements for software can dramatically increase the work effort and cost of software development; however, these requirements are often not identified until too late in the project to respond effectively. According to Barry Boehm: "A tiny change in NFR [non-functional requirements] can cause a huge change in the cost." [17]. He went on to cite the tripling of a $10 million project to $30 million when the response time (of an NFR) went from four seconds to one.

Step 5: Customer Issues Request for Proposal

In this step, the customer—with the assistance of the scope manager—prepares the request for proposal and evaluation criteria for the program and then issues it to a set of software suppliers. The *laundry lists* of the high-level functional and quality requirements (apportioned by subproject) are included as attachments to the request for proposal in order for suppliers to submit unit pricing estimates.

Step 6: Customer Selects Supplier Based on Submitted Unit Cost per FP

The scope manager assists the customer to evaluate the supplier responses and provides high-level input regarding the validity of their rates (based on industry rates). The customer selects one or more suppliers to meet the program needs. More specifically—for those parts of the work where the development or enhancement of software functions are involved—the pricing is provided by the supplier(s) in dollars per FP. For other parts, such as migration or operational upgrades, the pricing is provided in dollars per hour (or other appropriate units). The scope manager can provide a reasonability gauge for the cost per FP values by comparing the submitted rates using published project delivery rates in hours per FP (converted to dollars per hour).

Step 7: Requirements Specification Developed

This is the first step in northernSCOPE where the scope manager does not play an active part. The customer works directly with the supplier(s) to develop and *flesh out* the requirements for the subprojects.

Step 8: Scope Manager Baselines Software Size (in FPs) and Product Development

The requirements documents for those subprojects for which functional size is appropriate are the input for this step. The scope manager reviews the requirements and measures the baseline functional size in FP for each. This step is similar to finalizing the size of a floor plan for a building[5], and becomes the base against which any changes and progress are tracked.

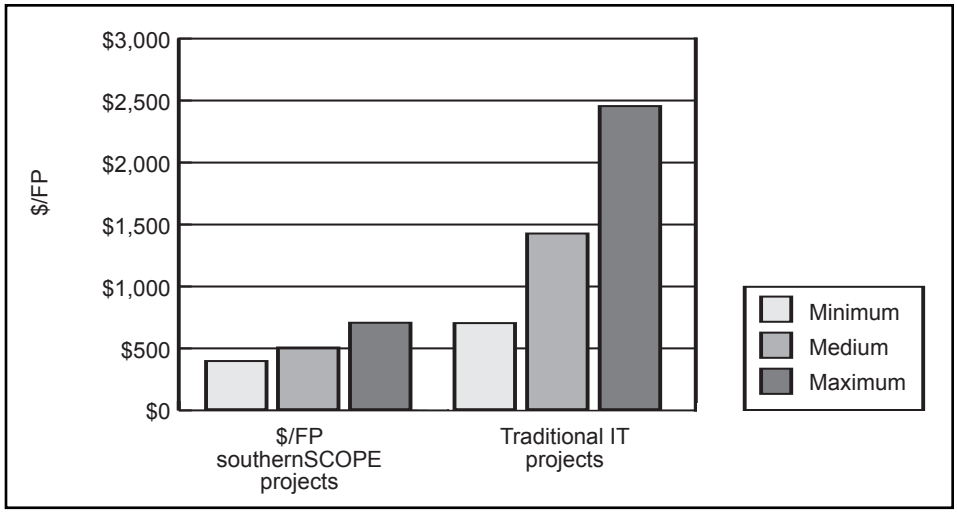


Figure 4: *Cost of southernSCOPE Projects Compared to Traditional IT Projects* [15]

Step 9: Scope Manager Sizes Project Changes and Cost Impact Is Evaluated

As changes are proposed for the project—by either the customer or the supplier in agreement with the customer—the scope manager collects and then records the data. A cost estimate is made of the cost and schedule impact of such changes at the point in the project where they are proposed and accepted (based on the unit cost[s] originally quoted by the supplier). A formal change management process facilitates this step.

Step 10: Scope Manager Quantifies Progress[4]

As each project progresses, the scope manager receives documentation from the supplier(s). Using the baseline(s) from step 8, the scope manager records the progress and prepares a formal status report for the customer. Usually this occurs on a monthly basis or on a sprint basis in Agile development, but the exact time frame of this reporting is established with the parties after step 7.

Step 11: Project Finishes and Customer Pays Supplier Based on FP Delivered

This is the second and final step of northernSCOPE where the scope manager is not directly involved. The customer pays the supplier(s) based on the FP delivered for each subproject and on whatever other mechanism where units of payment are used for the non-FP countable subproject(s). From the customer and supplier points of view, the program of work is now complete and the project manager closes the project(s).

Step 12: Experience Data Collected and Stored[4]

The scope manager finalizes the project by recording the data collected during and at completion of the project(s). Actual values for work effort and related project variables are recorded along with relevant project attributes.

It is worth noting that there are several concepts introduced in the scope management processes that are not traditionally included in ICT projects. These include:

- Analysis and classification of requirements into independently managed projects (or subprojects).
- Functional size measurement of the software requirements document and project scope.
- Baselining the project metrics.
- Estimating the project effort, duration, and cost based on historical *project actual* values.
- A feedback loop to estimate—and then incorporate and track—accepted changes into the existing project documents.

Through careful attention to project scoping and its management throughout the project, customers and suppliers alike can better specify, build, and acquire quality software products. After the completion of the entire project (i.e., after all steps of southernSCOPE are completed), then the results are tallied and the overall project is compared to those projects where no scope management was involved. Since the introduction of the concept, practitioners involved in southernSCOPE report substantially lowered costs per software functional size (Figure 4).

Coming Events

March 1-4
Ground System Architecture Workshop
Manhattan Beach, CA
http://csse.usc.edu/gsaw

March 3-5
Secure IT Conference 2010
Los Angeles, CA
www.secureitconf.com

March 9-12
23rd Annual IEEE-CS Conference on Software Engineering Education and Training
Pittsburgh, PA
http://cseet2010.dei.uc.pt

March 23-24
GovSec/U.S. Law 2010
The Government Security Expo & Conference/U.S. Law Conference & Exposition
Washington, D.C.
http://govsecinfo.com/events/govsec-2010

April 12-14
7th Annual International Conference on Information Technology
Las Vegas, NV
www.cis.gvsu.edu/~nandigaj/SoftwareEngineeringEducationTrack.html

April 26-29
22nd Annual Systems and Software Technology Conference
Salt Lake City, UT
www.sstc-online.org

April 27-30
MARTS 2010:
The Capacity Assurance Conference
Chicago, IL
www.martsconference.com

Coming Events: Please submit coming events that are of interest to our readers at least 90 days before registration. E-mail announcements to: <marek.steed.ctr@hill.af.mil>.

How to Apply Solid Scope Management for Success on ICT Projects

The five processes involved in the FiSMA Scope Management concept (Figure 2) are integrated concepts; however, all steps are not mandatory on every ICT project. The initiation and estimation steps are prerequisites for the other three components, and these subsequent steps are independent of each other. FiSMA recommends that organizations at least examine the last three steps for their applicability, but understands exceptions and lighter application needs exist [11]. Organizations that will benefit most from northernSCOPE are those representing business areas such as banking, insurance, and public administration because they are routinely involved in software acquisition and procurement. In these organizations, a core business is information management, and the business development centers on developing information systems and software. As such, traditional project management practices have proven to be insufficient for ongoing project governance because scope management is not considered until there is a project already underway. Even then, the project manager has more critical tasks to manage than those related to scope.

In addition to software acquirers, professional software suppliers are in need of organizational-level processes to support continuous process improvement and organizational learning. All supplier organizations could benefit from applying the northernSCOPE processes at both the project and organizational levels. Can every company gain from implementing this concept? Our experience bears out that there are some small- and medium-sized suppliers whose process maturity is considered to be ad-hoc or initial, and would be better served by first concentrating on developing or improving core processes such as time recording and invoicing. Nonetheless, the division of projects into software projects to be independently worked can assist even the most disorganized or immature organizations in improving ICT project management.

Through international collaboration and consulting, we have found that northernSCOPE (and for that matter, software process improvement) just doesn't always succeed even with the best of intentions. Reasons for failure or partial adoption of sound ideas such as scope management can run the gamut between lack of support or understanding to internal sabotage by organizations weary from *whiplash* changes imposed by management. It is useful to consider the critical success factors of process improvement before launching new concepts. Sources such as [18, 19, 20, 21] outline these critical success factors.

Conclusion

Through the results of northernSCOPE and southernSCOPE, scope management processes are a proven means of leveraging and augmenting professional project management on ICT projects. With the current levels of project rework in the vicinity of 45 percent of development effort, our industry surely needs to increase its ICT project success, and one proven way was expounded on in this article.

ICT program recovery can greatly benefit from a solid 12-step program, starting with an admittance of the facts, determination, trust, continuous control, and professional guidance and support from a scope manager.◆

References

1. Christensen, David S. "An Analysis of Cost Overruns on Defense Acquisition Contracts." *Project Management Journal* 3 (Sept. 1993): 43-48.
2. The Standish Group. "New Standish Group report shows more project failing and less successful projects." Standish Newsroom – Chaos 2009. 23 Apr. 2009 <www.standishgroup.com/newsroom/chaos_2009.php>.
3. eGovernment Resource Centre – State Government of Victoria (Australia). "southernSCOPE – Avoiding Software Budget Blowouts." 6 July 2007 <www.egov.vic.gov.au/index.php?env=-innews/detail:m1816-1-1-8-s-0:n-832-1-0-->.
4. FiSMA. *For Better Management*. northernSCOPE brochure. 8 Apr. 2008 <www.fisma.fi/in-english>.
5. Dekkers, Carol, and Pekka Forselius. *Increase ICT Project Success with Concrete Scope Management*. Proc. of the PMI Global Congress 2007 – Asia Pacific. Hong Kong. 29-31 Jan. 2007 <www.compaid.com/caiinternet/ezine/forselius-scope.pdf>.
6. Forselius, Pekka, and Risto Nevalainen. *The Role of SCOPE Manager as Link Between Engineering and Management Processes*. Proc. of the EuroSPI 2002. Nuremberg, Germany.
7. Morris, Pam. *Metrics Based Project Governance*. Proc. of IWSM/MetriKon.

Berlin, Germany. Nov. 2004 <www.compaid.com/caiinternet/ezine/morris-metrics.pdf>.
8. Wright, Terry. "Controlling Software Development, A Breakthrough for Business." *Software Quality and Process Improvement*. Melbourne, Australia: Software Engineering Australia. Nov. 2000.
9. ISBSG. *Software Development Projects in Government ... Performance, Practices and Predictions*. 2005 <https://www.ifpug.org/about/SoftwareInGovernment.pdf>.
10. *Insight, the Army Software Metrics Newsletter*. Summer 2002.
11. Forselius, Pekka. *Effective Scope Management Concept – A Key to Success*. Helsinki, Finland: FiSMA, Aug. 2003.
12. Project Management Institute. *A Guide to the Project Management Body of Knowledge – PMBOK Guide*. Newtown Square, PA: Project Management Institute, 2000.
13. Dekkers, Carol, and Pekka Forselius. *Certified Scope Manager based on northernSCOPE*. Training brochure. Dec. 2007 <www.qualityplustech.com>.
14. Forselius, Pekka. *Divide et Impera – Learn to Distinguish Project Management From Other Management Levels*. Proc. of the Software Measurement European Forum. Rome, Italy: May 2005.
15. Forselius, Pekka, et al. *Program Management TOOLKIT for Software and Systems Development*. Helsinki: Talentum, 2008.
16. Bundschuh, Manfred, and Carol Dekkers. *The IT Measurement Compendium*. Berlin/Heidelberg: Springer-Verlag, 2008.
17. Boehm, Barry. Preface to *Requirements-Led Project Management – Discovering David's Slingshot* by Suzanne and James Robinson. New York: Addison-Wesley Professional, 2005.
18. Dekkers, Carol. "Unleash the POWER to Improve." *Software Quality Professional* 3.1(Dec. 2000): 48-51.
19. DeMarco, Tom. "Mad About Measurement" essay in *Why Does Software Cost So Much?* New York: Dorset House, 1995.
20. DeMarco, Tom, and Timothy Lister. *Peopleware: Productive Project and Teams*. New York: Dorset House, 1999.
21. Willman, Paul. "Organizational Change in the 1990s." Lecture. University of Limerick, Ireland: Oct. 1996.

Notes

1. More about successful use of northernSCOPE can also be found in [5] and [6] while more on southern-SCOPE can be found in [7] and [8].
2. Scope management is put in the middle of Figure 1, though [12] introduces knowledge areas in a different order originally. The order is not important, but here we want to emphasize the central role of scope management in software development.
3. At this point, customers may refer to this as the project; however, it is more likely an ICT program with several projects. See step 2.
4. This step is not a part of southern-SCOPE.
5. When a floor plan is initialized at the beginning of a home construction project, it becomes the master plan that the built area in square feet or square meters and any changes to the floor plan are measured against.

Software Defense Application

The Standish Group's 2009 CHAOS report [2] states that software project success—meaning an on-time, on-budget, and a fully functioning product—has slipped to 32 percent. This, coupled with the abundance of sourced software development and enhancement projects in the DoD, should signal the importance of program scope management to defense software managers. The biggest benefit to the defense community is in increased customer communication and traceability of the original request through the software-intensive systems projects.

About the Authors

Carol Dekkers is CEO of Quality Plus Technologies, Inc. She consults clients on how to maximize project success through realistic estimating, software measurement, and scope management. Dekkers holds designations as a Project Management Professional, Certified Management Consultant, Certified FP Specialist, Information Systems Professional, and Professional Engineer (Canada). She is known internationally as an expert, a speaker, and an author in software metrics, functional size measurement, project scope management, and IT globalization, and has served as a visiting scientist with the SEI. Dekkers is the co-author of several books, including: "The IT Measurement Compendium - Estimating and Benchmarking Success with Functional Size Measurement," "Program Management Toolkit for software & systems development," "IT Measurement: Practical Advice from the Experts," and "Practical Project Estimation."

Quality Plus Technologies, Inc.
8430 Egret LN
Seminole, FL 33776
Phone: (727) 393-6048
E-mail: dekkers@qualityplustech.com

Pekka Forselius is a business partner, CEO, and project management consultant at 4SUM Partners in Finland. Forselius developed the Experience Pro data-collection concept and was product manager for Experience Pro software. He is a researcher and developer of project management methods and concepts, including northern-SCOPE Scope Management and the FiSMA 1.1 ISO/IEC 29881 Functional Size Measurement standard. Since 2000, he has been the primary representative of the national body of Finland to the ISO/IEC JTC1/SC7 standardization working group WG12, Functional Size Measurement. Forselius is also the president of the international benchmarking organization (ISBSG). He is co-author of several books including "Program Management Toolkit for software & systems development," "Practical Project Estimation," and "Applied Statistics for Software Managers." Forselius holds an executive MBA and a master's degree in informatics.

4SUM Partners, Inc.
Tekniikantie 14
Espoo, Finland FIN-02150
Phone: +35 85 05160416
E-mail: pekka.forselius@4sumpartners.com

Stealth CPI: Managing Work Products to Achieve Continuous Process Improvement

Ron Abler and Ted Warren
NAVAIR

While identifying, defining, and organizing activities into graphical process diagrams is helpful during Continuous Process Improvement (CPI) planning, it's not very useful to our managers and their teams during implementation. Most managers understand—and therefore manage—through work products (WPs). Managing WPs has proven to be simpler and more intuitive than managing processes, dealing directly with the end-products themselves. Stealth CPI centers on existing WPs, permitting organizations to implement process improvement (PI) incrementally, purposefully—and even stealthily.

Most if not all PI standards, models, and methodologies concern themselves with processes during the planning and implementation phases. But when it comes to the evaluation phase, evaluators do not look at processes but at the concrete evidence (artifacts and WPs) showing that the processes have been followed. Nevertheless, PI remains an abstract exercise that is not intuitively obvious to the majority of technical professionals who are not specialists in the PI field. This fact has led to a myriad of models and methodologies that attempt to simplify the concept of PI. The end result has been the creation of an extensive consulting industry dedicated to shepherding organizations through the confusing maze and jargon of formal standards compliance.

In our organization, the Presidential Helicopters Program In-Service Integrated Product Team (VH-IS), we are prototyping a concept that we call Stealth CPI. Our goal is to improve our processes by structuring our WPs so that our people can be best practices- and CMMI-compliant just by doing their jobs. This concept is based on lessons learned from previous implementations of CMMI in other organizations:

1. Process diagrams are abstract and are, at best, graphic representations of activities conducted within and in support of the described process. Thus, process diagrams are tools, not products.
2. Processes are made up of activities, and each activity results in one or more WPs. Only the WP, not the generating activity, can transfer the activity's productivity elsewhere within an organization. Any activity or suite of related activities that does not produce a desirable WP is held suspect because it may be unnecessary or unproductive.

We feel that neither processes nor activities alone benefit organizations. With this in mind, our plan implements CPI by managing WPs instead of processes. Stealth CPI meets its PI goals in three phases:

1. **Planning Phase**
   - Documenting and refining the visual representation of the improved processes.
   - Mapping WPs to activities.
   - Developing work folders (all containing guidance) to support improved WPs.
2. **Implementation Phase**
   - Publishing the work folders.
   - Assigning the improved WPs.
3. **Assessment Phase**
   - Conducting continuous assessments. For example, SCAMPI B and C assessments in a CMMI environment.

Project managers are a breed apart in American business and government. They walk a narrow line defined by requirements, schedule, budget, and resources, complicated by the demands of superiors and the needs of subordinates. Happy is the manager who knows exactly what WPs are required and when, and who has been given all the resources necessary to produce those WPs correctly, completely, and in a timely manner. In fact, the same thing can probably be said of most employees in any organization, regardless of the organization's business.

When the specter of PI rears its intimidating head (as it will eventually in every competitive business), all too frequently the conventional response is to start with process definition in the form of process diagrams. In too many cases, the end result of this classic approach is improved process diagrams instead of improved processes. Giving project managers a process diagram and asking them to make the contained improvements is usually a waste of their time. Training project managers to become PI specialists—in the hope that the *worker bees* will fulfill the desires of the management queens and drones—can be expensive and wasteful. We believe it is more efficient and cost-effective to implement PI by:

- Rolling out improved WP tool sets rather than improved process diagrams.
- Training our people to be experts in our WPs rather than in PI.

And our people agree.

Stealth CPI is a *bottoms-up* approach that takes advantage of a simple fact: It is far simpler and infinitely more intuitive to identify and improve WPs than it is with processes. It is more conceivable to envision a successful organization without a single process diagram than it is to imagine one without WPs. Even the clearest diagram of the best possible process can't single-handedly improve an existing process. This is where PI models and methodologies normally enter the picture: to show us where to go next. Regardless of the standard to be used, all of them start with processes but end with WPs. The reason for this is that assessing the goodness of a process is neither feasible nor objective. Take the following exchange as a typical example:

Evaluator: "Do you conduct peer reviews?" (If peer reviews are part of the standard being evaluated, then "Yes" is the only correct answer to this question).

Evaluee: "Yes."

Evaluator: "Prove it!"

At this point, it is neither correct nor sufficient to reply: "Our next peer review is in two weeks. Come back then and watch."

In other words, the proof lies not in the process, but in the artifacts and WPs that result from the process. Evaluators routinely presume that the existence of a desired artifact proves that its generating activity did, in fact, take place. The quality of the artifact is presumed to reflect the quality of its generating activity.

NAVAIR has developed its "System

Engineering Guide," which mandates a process of System Engineering Technical Reviews (SETRs) [1]. The SETR entry and exit criteria are more easily understood when they are described as WPs than when they are portrayed as activities. Thus, a standardized report containing the output of the review activities usually makes a more useful exit criterion than a statement that the reviews activities were completed.

Since evaluations are always conducted at the artifact level, it makes sense to approach PI from the same perspective. PI constitutes a chain with connected links from the envisioned process through the implementing activities right down to the artifacts that constitute the WPs. It is easier, more intuitive, and more operationally effective to pull the chain by improving real-world WPs than it is to push the chain by improving abstract processes.

The Capability Waypoint Matrix

Robert Jacob, the Head of the Aviation Safety Department at the Atlantic Test Range at the Naval Air Station (NAS) Patuxent River, and Ron Abler developed the concept of displaying and tracking progress in the department's implementation of CMMI with a two-dimensional matrix, called a Capability Waypoint Matrix (CWM). Abler presented the CWM at a Software Engineering Process Group (SEPG) [2] meeting where the SEI had solicited ideas for simplifying the CMMI. A panel of CMMI sponsors, CMMI instructors, and SCAMPI leads peer-reviewed the CWM and declared it to be CMMI-compliant.

The Aviation Safety Department rolled out their CWM in October, 2007, and in 2008 published their T-Risk model [3], which was the first known implementation of CMMI in an operational aviation safety environment.

As the term *matrix* implies, a CWM is simply an array of rows and columns (see Figure 1). Each cell that is defined by the intersection of a single row and column is called a waypoint. The CWM is a snapshot of the work-product improvement status at a specific moment in time. In aviation, a waypoint is a point in physical space defined by latitude, longitude, altitude, and time of passage. In the CWM, a waypoint is defined by process name, effectiveness, efficiency, and date/time of the snapshot (i.e., evaluation). The rows and columns arrange the WPs in the structure defined by the CPI model in use.

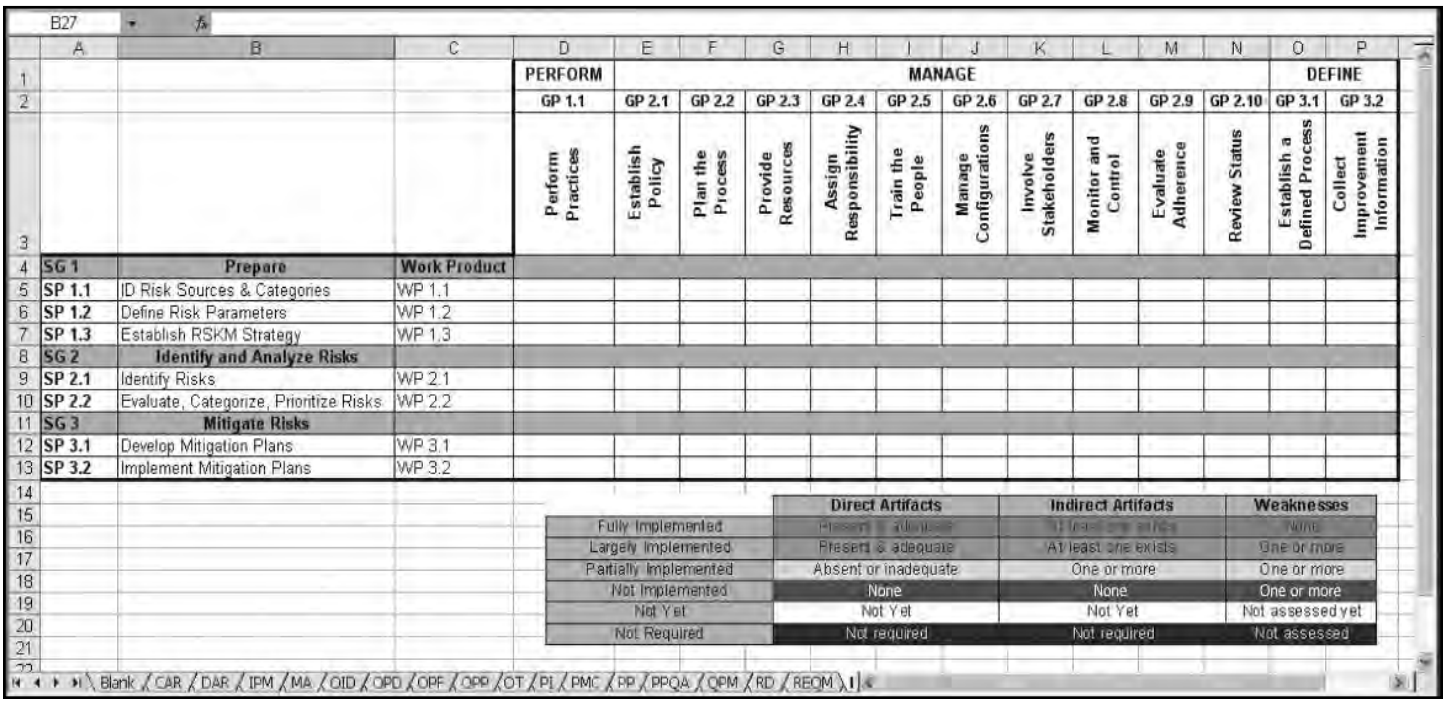
While it is true that the CWM can be used to support any PI methodology that has artifacts (or WPs), the following description is based on the CMMI model [4], which is the methodology that VH-IS uses.

A CWM is generated as a spreadsheet for each CMMI process area in the organization's profile. Figure 1 depicts a simplified CWM for the risk management process area. The Y-axis (represented by the rows) charts overall effectiveness, and the X-axis (represented by the columns) charts overall efficiency. In CMMI, effectiveness (defined as simple performance) is defined by the achievement of specific goals (SGs). Efficiency (i.e., better, faster, cheaper, smarter) is defined by fulfillment of the generic goals. Therefore, in the CWM, the rows are populated by specific practices (SPs) and the columns by the generic practices (GPs). Thus, columns A and B of this CWM hold the risk management SGs and practices, while rows D through P hold the generic practices. Column C holds the WP (i.e., direct artifact) that results from each specific practice (these are what the SCAMPI team looks for). Since Stealth CPI tracks and manages WPs rather than processes, it is important that each row hold only one WP. If a SP generates more than one WP, add a row or rows as necessary to accommodate them. WPs tend to be organization-specific, so Figure 1 refers to them generically as WP 1.1 through 3.2.

Examples of Risk Management WPs might be a risk list for WP 2.1 and a fully categorized and prioritized risk list for WP 2.2. WP 3.1 would obviously be mitigation plans. If there are more than one, they can be designated as 3.1a, 3.1b, etc. WP 3.2 might be periodic reports that track and document the implementation of each mitigation plan.

Once a CWM has been completed to this point (filled in with SPs, GPs, and WPs), the simple secret of Stealth CPI becomes apparent. It is so simple that it sounds trivial. All that is required to achieve PI is to apply each GP, one at a time, to each WP and track the status of so doing in each waypoint. That status can be represented textually (e.g., excel-

Figure 1: *Sample Capability Waypoint Matrix (for the Risk Management Process Area)*



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1				PERFORM	MANAGE										DEFINE	
2				GP 1.1	GP 2.1	GP 2.2	GP 2.3	GP 2.4	GP 2.5	GP 2.6	GP 2.7	GP 2.8	GP 2.9	GP 2.10	GP 3.1	GP 3.2
3				Perform Practices	Establish Policy	Plan the Process	Provide Resources	Assign Responsibility	Train the People	Manage Configurations	Involve Stakeholders	Monitor and Control	Evaluate Adherence	Review Status	Establish a Defined Process	Collect Improvement Information
4	SG 1	Prepare	Work Product													
5	SP 1.1	ID Risk Sources & Categories	WP 1.1													
6	SP 1.2	Define Risk Parameters	WP 1.2													
7	SP 1.3	Establish RSKM Strategy	WP 1.3													
8	SG 2	Identify and Analyze Risks														
9	SP 2.1	Identify Risks	WP 2.1													
10	SP 2.2	Evaluate, Categorize, Prioritize Risks	WP 2.2													
11	SG 3	Mitigate Risks														
12	SP 3.1	Develop Mitigation Plans	WP 3.1													
13	SP 3.2	Implement Mitigation Plans	WP 3.2													

	Direct Artifacts	Indirect Artifacts	Weaknesses
Fully Implemented	Present & adequate	At least one exists	None
Largely Implemented	Present & adequate	At least one exists	One or more
Partially Implemented	Absent or inadequate	One or more	One or more
Not Implemented	None	None	One or more
Not Yet	Not Yet	Not Yet	Not assessed yet
Not Required	Not required	Not required	Not assessed

lent, good, inadequate), numerically (e.g., 1 through 10), or graphically (e.g., green, yellow, and red), as desired. The CWM in Figure 1 uses colors to denote status in a manner similar to the SCAMPI.

The technique of Stealth CPI starts with the waypoints in column D. If it can be demonstrated that a WP simply exists and that its indirect artifacts (e.g., memos, meeting minutes, emails, etc.) demonstrate that it is in regular use, then its waypoint in column D can be filled in with a satisfactory status. Once all the SPs in a process area have satisfactory marks in column D of the CWM, it can be said that the organization is minimally effective in that process area (i.e., Capability Level 1 "Performed").

Admittedly, higher capability levels are more difficult to achieve than merely demonstrating the existence of WPs, but the Stealth CPI technique remains just as simple at all capability levels: Apply each GP, one at a time, to each WP and improve that WP as necessary to comply with the GP. Take, for example, a mitigation plan, WP 3.1 (in row 12, column C). The waypoint under GP 2.5, Train the People (row 12, column I) would contain a satisfactory mark if it can be shown that the employee(s) who develop mitigation plans are properly trained to do so. Similarly, the waypoint under GP 2.6, Manage Configurations would hold a "Not Applicable" mark if the mitigation plan was not a configuration item (CI). Since it is likely that a mitigation plan would be a CI, the waypoint in column J would hold a satisfactory grade if the mitigation plan shows evidence of configuration management, such as a current document version number.

One can see that the CWM tracks both the simple existence (effectiveness) and the goodness of that WP (efficiency) relative to the discriminators required by whatever standard an organization chooses to use (such as the GPs in CMMI). It is the CWM that enables Stealth CPI to achieve PI, one WP at a time.

Work Folders

Once the processes, their related activities, and the resulting WPs have been mapped onto the CWM, it becomes possible to begin improving processes by directing the desired improvement in the organization's WPs.

Take GP 2.6, Manage Configurations, for example: Every WP (i.e., each row) in the CWM has its waypoint in the GP 2.6 column. If a WP is a configuration item (i.e., appears on the configuration item list, itself a WP), then Stealth CPI expects that evidence of configuration management will be reflected in the WP. In the case of a recurring report, the *improvement* might be the simple addition of a document version or control number. Improvement in WPs are managed through the use of work folders, which completely define the *who*, *what*, *when*, *where*, *why*, and *how* of each WP.

The goal of a work folder is to provide staff members with all the information and tools necessary to produce WPs of the desired quality. As appropriate to each WP, a work folder may contain a template, a tailoring guide, criteria for use, a process or activity diagram, complete instructions (if not contained in the template itself), perhaps a PI Indicator Description, and even the actual training materials (or pointers to the training) necessary for proper use and delivery. In a nutshell, the work folder completely defines the activity or activities that generate the WP. In total, the work folders implement the processes of the organization in an immediately usable form.

"Our staff does not have to be specifically trained in CMMI or even conversant with PI theory, concentrating on doing what they do best: generating the highest-quality WPs possible."

Stealth CPI in the Program Office

VH-IS is adopting the Program Management Office (PMO) approach of consolidating functions and standardizing methodologies and templates for use in the management and delivery of projects. This new approach will be the prototype for Stealth CPI. If successful, Stealth CPI will be implemented for new projects and eventually retrofitted throughout the organization. Stealth CPI will be the basis for re-use, wherever possible, of PMO-like processes and WPs. VH-IS has selected the continuous representation of CMMI with a tailored profile consisting of six high-priority process areas: project planning, project monitoring and control, requirements development, requirements management, configuration management, and risk management.

The first step was to generate a CWM for each process area by listing the SPs in the rows, and the GPs in the columns. The next step was to map the WPs from NAVAIR's SETR (entry and exit criteria) to their related SPs. This involved adding some practices or activities to account for non-CMMI WPs required by the SETR as well as adding some new WPs expected in the CMMI but not called for in the SETR.

After each individual CWM has been fully populated (such that every WP maps to a single practice or activity and every practice or activity maps to a unique WP), the work folders for each WP can be created and placed in a shareable repository.

Once a work folder has been completed and made available to staff members on a project-by-project basis, the VH-IS can track, manage, and ultimately fulfill all of the requirements of the governing SETR in a CMMI-compliant manner by simply assigning the WPs that fulfill the entry and exit criteria of the SETR (plus supporting CMMI-specific WPs, as necessary). From this point on, our ultimate goal of constant PI can be achieved through regular WP improvement as long as each improvement is defined and facilitated by the tools in the concomitant work folders. Our staff does not have to be specifically trained in CMMI or even conversant with PI theory, concentrating on doing what they do best: generating the highest-quality WPs possible.

In summary: Contrary to classic CPI, we will employ and improve our existing WPs; established processes and their improvement will be the consequent by-products. The CMM will serve as our tool and our map.

Early Success Indicators

At VH-IS, we are on the verge of declaring success at Capability Level 1 for our profile. As soon as we have mapped an existing WP (direct artifact) to each SP and can demonstrate their active use with indirect artifact(s), we can claim Level 1 because we can prove that we are performing each specific practice. Considering that, we will have reached Level 1 without requiring the very time-consuming and (unfortunately) distracting SEPG. In fact, we will have achieved

this with only the part-time involvement (16 hours per week) of the standards engineer and a couple of hours per week of the team lead's time—a definite cost-avoidance. Furthermore, we have gotten this far without requiring any formal CMMI training of our staff—a definite cost-savings. This is an early indicator that we can continue this technique through Levels 2 and 3, and we should be able to do so very cost-effectively, while imposing little or no process-improvement distraction into the equation.

Also, in the course of reaching Level 1, we will have mapped our SPs to the NAVAIR-mandated technical review system, thereby killing two birds with one stone: We have demonstrated the feasibility of implementing a technical review system that is intrinsically CMMI-compliant through common WPs. We consider this to be a necessary milestone toward the success of our initially stated ultimate goal, which is to structure our WPs so that our people can be best-practices compliant just by doing their jobs.

Ten Steps to Successful Stealth CPI

If you plan on starting to utilize Stealth CPI in your organization, here are 10 essential steps to success:

1. Create a CWM template for each process area[1].
2. List each activity within the process on its own row in column A and B. In CMMI, these are SPs.
3. List the title of each activity's WP in column C.
4. If an activity generates more than one WP, add a row for each WP, repeating the activity name in columns A and B, and the name of the artifact in column C. In column D, insert a word or phrase that will document the simple existence of each artifact.
5. Determine the discriminators that will be used to evaluate the incremental improvement of the process. In CMMI, these are the GPs. Place the name of each discriminator in the top row of each column, beginning with column D.
6. For each WP, assign an owner/manager who is responsible for its maintenance and upkeep.
7. Define a complete work folder for each artifact or group of artifacts that represents a discrete WP.
8. Conduct internal audits (gap analyses) of the artifacts and WPs, recording the status of each in the appropriate waypoint, starting with column D (*exists*) and denoting each qualifying waypoint under the discriminators, as appropriate.
9. Use the results of the audits to evaluate the overall PI program and to continue improving the organization's WPs.
10. When all waypoints in a process are satisfactory for the desired next level of improvement, DECLARE VICTORY!◆

Software Defense Application

Defense organizations will find that Stealth CPI is much more intuitive and cost-effective than traditional CPI because it approaches the problems from the perspective of real-world work products rather than abstract processes. This article steps the reader through the set-up and implementation of Stealth CPI with the use of its central tool, the Capability Waypoint Matrix. Stealth CPI saves time and money over more conventional approaches because it requires less process-specific training, fewer people dedicated to the CPI project itself, and therefore less overhead to achieve the same results. Better-managed WPs foster better reuse, result in more efficient implementation, and are easier to improve than abstract processes.

References

1. Commander, Naval Air Systems Command. *NAVAIR Systems Engineering Technical Review Process*. NAVAIRINST 4355.19D. 17 Apr. 2009 <http://nawctsd.navair.navy.mil/Resources/Library/Acqguide/4355-19.doc>.
2. Abler, Ron. *Stealth CMMI*. Proc. of the SEI/NDIA CMMI V. 1.2 and Beyond Workshop. Montreal, 2007.
3. Jacob, Bob, and Ron Abler. *T-RISK – Risk Management for Test Range Safety*. Atlantic Test Range, Naval Air Systems Command. Patuxent River, MD. 17 Mar. 2008.
4. Chrissis, Mary Beth, Mike Konrad, and Sandy Shrum. *CMMI: Guidelines for Process Integration and Product Improvement*. 2nd Ed. New York: Addison-Wesley, 2007.

Note

1. Contact Ron Abler by e-mail for free-of-charge, ready-to-use sets of CWMs.

About the Authors

Ron Abler is a standards engineer for Sabre Systems Inc., serving the VH-IS's Independent Verification and Validation for NAVAIR at the NAS – Patuxent River, Maryland. He also is a Best Practices Champion, CMMI consultant, and SEI-authorized CMMI Instructor. He served as the U.S. Navy's program manager for office automation and, after retirement from the Navy, as the vice president of AOM Associates. Abler led the USAF's first successful CMM Level 2 appraisal, chaired four CMM SEPGs leading to SCAMPI appraisals, was the independent standards compliance manager for three NAVAIR teams.

Sabre Systems, Inc.
48015 Pine Hill Run RD
BLDG 3
Lexington Park, MD 20653
Phone: (301) 995-3873
E-mail: rpabler@atlanticbb.net

Ted Warren is the Independent Verification and Validation team lead and the risk manager for the VH-IS for NAVAIR at the NAS – Patuxent River, Maryland. He has 28 years of experience in systems engineering, systems analysis, and program management. Warren earned a bachelor's degree in geology from Dickinson College and a master's degree in systems engineering from Johns Hopkins University. He is also an instructor for the Master of Science in Systems Engineering Program at Johns Hopkins.

NAVAIR
48100 Shaw RD
RM 2A01
Patuxent River, MD 20670
Phone: (301) 757-1825
E-mail: heath.warren@navy.mil

CMMI, Swiss Cheese, and Pareto

Darrell Corbin
Independent Consultant

Getting ready for a CMMI appraisal is rarely easy. But what if your organization is virtual, has critical 24/7 commitments, and overtime is the norm? What if key staff is overloaded and time available for CMMI is minimal? This article provides a case study showing how these obstacles can be overcome in preparation for a CMMI for Development (CMMI-DEV) Level 3 appraisal.

Imagine that you've just been asked to assume the role of process analyst and lead the effort for an organization to pass a CMMI-DEV Level 3 appraisal. Most of the 40-member staff has participated in earlier self assessments and the software developers are from a Level 5 company. Everyone knows about CMMI and its predecessors, primarily the Software CMM. The assignment sounds easy, right? Wrong!

While I've come across many organizations with similar challenges, this article will focus on a case study of one organization—for our purposes, I'll call them Company C—that overcame obstacles synergistically using the *Swiss Cheese Method* and the Pareto Principle.

The Work Environment

Company C's IT staff is mostly virtual with members in three states. The developers (subcontractors) are from India. The group is responsible for key company software and must be available 24/7. Overtime is the norm. Software upgrades are frequent and take priority. Time available for CMMI is rare. Meetings are held using WebEx, and deliverables are stored on SharePoint, Wikis, file servers, or laptops. There is no Process Asset Library (PAL) at the beginning.

Company C has improved software processes for years by facilitating workshops. Conference rooms are lined with flip charts showing process flows, data stores, artifacts, and more. But this approach doesn't work when the staff is absent or virtual, and the IT staff has only a few minutes available to help leadership at random times. What did Company C do?

Remember the Swiss Cheese Method?

Alan Lakein's 1973 book, "How to Get Control of Your Time and Your Life," introduced the Swiss Cheese Method and talked about how we can nibble away at a task or project. He suggested that when we only have five minutes or less, we should spend it on high-priority tasks and constantly nibble away at them. Otherwise, we slide the schedule and keep waiting until we have *enough* time to work on a task.

Project plans are rarely defined in minutes and doing so isn't recommended. In the case of Company C, the process analyst used the Swiss Cheese Method on critical path tasks. The process analyst did not get an hour of a functional analyst's time, but was given a few minutes after specifically detailing the help that was needed.

By using the Swiss Cheese Method in a variety of ways, Company C was able to:

- Identify source documents (versus multiple versions) for their PAL. More than 100 documents are put in the PAL.
- Obtain comments on sections, one at a time, of their quality plan and dozens of other process documents.
- Prioritize process improvement team actions. The team completes more than 100 actions prior to the appraisal.
- Resolve inconsistencies in documents under configuration management.
- Update portions of the project plan.
- Resolve dozens of internal assessment comments (one at a time).

Tools such as instant messaging helped identify these small blocks of time, even as little as five minutes. In effect, this allowed for the constant nibbling away at the hundreds of mini-tasks required to get ready for the appraisal. The process analyst worked with the staff to identify these small time slices every day and progress was closely measured.

Use of the Swiss Cheese Method required the process analyst to work much closer with the project manager as compared to a typical project. MS Project was used to develop the initial schedule, then Primavera P3e was used to manage the CMMI project. That's where another time-management concept, the Pareto Principle, came in handy.

Thank You, Vilfredo Pareto

Many of us have used the 80-20 rule, the more common term for the Pareto Principle. Recall that Vilfredo was an Italian economist who found that 80 percent of the land in Italy was owned by 20 percent of the population. His findings evolved into the rule that says, by one application: 20 percent of the time spent on the *vital few* yields 80 percent of the results—and 80 percent of the time spent on the *trivial many* yields only 20 percent of the results.

So what does that have to do with CMMI? When Company C's CMMI effort was started, there was no project plan, no PAL, few completed process documents, and multiple versions of key documents. And there was no common understanding of the definition of all 163 key areas (22 process areas, three defined by Company C). From a Pareto view, 80 percent of the time spent on process activities only yielded 20 percent of the target: 100 percent CMMI compliance.

Prior to the CMMI project, there was little evidence—by any measure—that the time was being well-spent. Vilfredo would be happy to know that Company C used his principle to help them focus on the *vital few* to obtain the largest portion, the 80 percent, of the positive results.

The process analyst and project manager reviewed the CMMI statement of work and associated work breakdown structure and schedules, and identified tasks in the *vital few* category, including:

- Updating the quality plan. This turned out to be a major task.
- Identifying metrics for each process area: to give proof that use of CMMI was yielding positive, measurable results in terms of cost, quality, and schedule.
- Establishing a quality audit program and conducting quality audits.
- Training for all Level 3 and some Level 2 Process Areas (presented in one-hour segments via WebEx).
- Updating self-assessment by using a proprietary spreadsheet for the staff to record their personal assessments by key area.

Software Defense Application

The article describes some creative techniques defense organizations can use to achieve 100 percent CMMI compliance in a virtual environment. This case study of a major contractor describes how the organization prepared for a CMMI-DEV Level 3 appraisal. Two of the techniques come from basic time management tenets: the Swiss Cheese Method and the Pareto Principle. Other lessons learned addressed the process improvement team, configuration management, and communications techniques.

Synergy of Swiss Cheese and Pareto

Tasks are constantly reprioritized in any big project, but determining which tasks are part of the *vital few* may not be an easy matter. Also, assignments are made and commitments agreed to throughout a project. But in Company C's production system, firefighting, frequent block points, and resultant overtime caused havoc with the CMMI schedules.

But judicious use of the Swiss Cheese Method and the Pareto Principle combined to help attain full CMMI compliance (Company C's CMMI appraisers called it *validation*) prior to the appraisal. Instead of juggling three or four balls, there were dozens of balls in the air at any one time.

Lessons Learned

Company C's experience can be applied to others planning a CMMI adventure. Here are some other lessons learned:

- Since onsite process *workshops* weren't feasible, virtual workshops were held. Instead of flip charts on the wall, the participants used a one-page graphic called the *project process flow*. This complex diagram showed every step of the project using Project Management Body of Knowledge categories such as planning, executing, controlling, and closing. Company C's dozens of process documents (completed and in-process) were identified. A sub-group was assigned to update the software development life cycle processes (another non-trivial effort).
- Throughout the project, a CMMI appraiser was involved. Periodic *validations* were performed to identify areas in compliance and those needing either improvement or time to become institutionalized.
- The list of participants in the appraisal constantly evolved. The Process Improvement Team (PIT) members negotiated the list with management to help assure that all job categories were adequately represented.
- PIT membership was also frequently negotiated with management. There are different types of analysts, developers, and support personnel. The scope of the PIT also evolved and helped focus on the *vital few* actions.
- IBM's Rational ClearQuest software was used for change requests and Rational ClearCase for deliverables (requirements, design, code, etc.). Getting these under configuration management was not a trivial effort due to multiple versions on file servers and MS SharePoint.

Process improvement is difficult work, including getting ready for a CMMI appraisal. But by using basic time management techniques, persistence, and creativity, organizations can maximize their probability of success. Lessons learned should be documented and shared with others. As the saying goes, "We're all in this together."◆

About the Author

Darrell Corbin is a process consultant and has been helping organizations implement CMMI and its major predecessor, Software CMM, since 1992. He retired from Boeing in 2006 where he was an associate technical fellow/process engineer, and helped them deploy software process improvement. While at Boeing, Corbin received Software Engineering Process Group certifications, and was a second management coach for Fujitsu's Macroscope Methodology. He has been an invited speaker at more than 100 conferences, professional society meetings, and universities, and has published 19 articles.

1226 65th ST SE
#101
Auburn, WA 98092
Phone: (253) 735-6368
E-mail: dcorbin@
darrellscorbin.com

Web Sites

Information Technology Infrastructure Library (ITIL)

www.itil-officialsite.com

After reading Jeffrey L. Dutton's article, explore one of the components of his integrated framework to process improvement. This official Web site for the ITIL offers a comprehensive host of resources for ITIL users. Learn about ITIL's benefits in regards to costs, capabilities, and customer satisfaction; read testimonials, white papers, and case studies from users and organizations worldwide; and learn everything about ITIL certification from training to exams to what it takes to earn the ITIL Expert Level. There is also the recent "refresh" from ITIL Version 2 to Version 3: The site navigates readers through the differences between the two versions, discusses Version 2's future, and outlines the benefits of Version 3.

Webinar: CMMI V1.3 Product Suite

www.sei.cmu.edu/library/abstracts/webinars/10feb2009.cfm

A perfect supplement to Mike Phillips and Sandy Shrum's *Process Improvement for All: What to Expect from CMMI Version 1.3*, this webinar session discusses CMMI V1.3. Mike Konrad, manager of the CMMI Model Team and CMMI Chief Architect, and Rusty Young, manager of the SCAMPI Appraisal Team, focus on updates to the suite: clarity of high maturity; more effective generic practices; appraisal efficiencies; and commonality across the constellations of Development, Acquisition, and Services.

Information and Communications Technology (ICT) Project Management Practitioner Network

http://ictpmpractitioner.ning.com/

After reading Carol Dekkers and Pekka Forselius' article on ICT program recovery, you may want to join the free online learning network designed for both beginners and advanced practitioners of project management in ICT solutions development. Users freely initiate discussion and present best practices on how to effectively coordinate ICT initiatives in government, education, people organization, and small and medium enterprises. Users can also customize their own pages with discussion forums, blogs, and video to facilitate what the ICT Project Management Practitioner Network calls "the open online discussion and sharing of the practitioner's contextual application of project standards and ICT solutions development."

DoD Continuous Process Improvement Transformation Guidebook

https://acc.dau.mil/CommunityBrowser.aspx?id=32364

Ted Warren and Ron Abler introduced readers to a "stealth" method of achieving Continuous Process Improvement through work product management, now explore the guide used by the DoD to design and manage Continuous Process Improvement efforts. The guide standardizes terminology and incorporates best practices from the experiences of both defense organizations and leading private industries. It is a strategic approach for developing a culture of continuous improvement in the areas of reliability, process cycle times, and costs in terms of less total resource consumption, quality, and productivity. The guide provides a framework that is used for implementing and sustaining a culture of continuous improvement, focusing on four key elements: a broad-based and structured Continuous Process Improvement implementation method, aligned goals, project management and implementation, and providing a framework and useful checklists to gauge organizational maturity in using Continuous Process Improvement.

CMMI Appraisals

www.sei.cmu.edu/cmmi/tools/appraisals

Darrell Corbin's *CMMI, Swiss Cheese, and Pareto* explained how to prepare for an CMMI appraisal—now go the SEI's official source for information on the CMMI Appraisal Program. The Web site details the Appraisal Requirements for CMMI, including the characteristics of SCAMPI at Class Levels A, B, and C. There are also resources including publications and presentations related to appraisals, frequently asked questions, official communications directly from the SEI's Appraisal Program, an SEI Partner directory and network, and a link to published appraisal results.

CMMI or Agile: Why Not Embrace Both!

www.sei.cmu.edu/reports/08tn003.pdf

As an exceptional companion piece to his article *Love and Marriage: CMMI and Agile Need Each Other*, Hillel Glazer (this time with co-authors Jeff Dalton, David Anderson, Mike Konrad, Sandy Shrum) explore the two methods often at odds with each other. Their report clarifies why the discord need not exist and proposes that CMMI and Agile champions work toward deriving benefits from using both and exploiting synergies that have the potential to dramatically improve business performance. The authors outline the two primary reasons for the discord, examine the divergent origins of both methods, analyze and debunk the factors that have led to the negative perception, detail "truths" about the two methods, show the value of both through analysis and comparison, and even provide a "call to action" for both CMMI and Agile experts.

CrossTalk: Looking Back at CMMI

www.stsc.hill.af.mil/crosstalk/2007/02/
www.stsc.hill.af.mil/crosstalk/2006/04/
www.stsc.hill.af.mil/crosstalk/2002/02/
www.stsc.hill.af.mil/crosstalk/2000/07/

If you're new to CrossTalk, you may have missed past issues that also explored the famed process improvement approach. The February 2007 edition explored the changes in Version 1.2, Appraisal Method for Process Improvement B, and how to achieve a CMMI Level 2 rating within six months. You can also see if Watts S. Humphrey, Dr. Michael D. Konrad, James W. Over, and William C. Peterson were right with predictions in their article *Future Directions in Process Improvement*. April 2006, themed *Alternate Mixes for CMMI*, included discussions of combining Agile and traditional software development techniques, managing cultural changes, and performing cooperative appraisals. Issues dedicated to CMMI also reach back early in the decade, with February 2002 simply themed *CMMI* and July 2000 focused on *Process Alchemy* in CMMI.

Love and Marriage: CMMI and Agile Need Each Other

Hillel Glazer
Entinex, Inc.

Agile's values and practices ensure critical, long-term process success, making it an ideal partner of the CMMI[1] framework, which delivers a robust infrastructure of organization-wide, broadly inculcated continuous improvement and optimization. Intended for an audience of process improvement professionals, CMM left out some of the basic elements critical to long-term process success [1] *that—as luck would have it—Agile values and practices supply. Together, Agile and CMMI complete each others' capabilities and can lead to fast, affordable, visible, and long-term benefits.*

In early 2001, the Agile movement can be arguably said to have gelled with the formulation of the "Manifesto for Agile Software Development" (often called the Agile Manifesto) [2]. Other than two items published later that year—one in CROSSTALK [3] and another in *IEEE Software* [4]—much of the writing from 2001-2008 on the topic of CMM [5, 6] (or CMMI) and Agile development had been limited to online sources such as e-mail groups, user forums, blogs, and wikis[2]. In that time, much of what was written on the topic was mostly on how the two bodies of ideas were incompatible.

Barry Boehm and Richard Turner's "Balancing Agility and Discipline" [7] provides a sound, practical, robust risk-based approach to reconciling what was widely perceived as being orthogonal interests of discipline, à la CMMI (then v1.1) and *agility* (of any variety). Nonetheless, the *process myth* of irreconcilability between CMMI and Agile persisted. User-group-type organizations—such as the SEI's Systems and Software Process Improvement Network and Agile Project Leadership Network (APLN) groups—were fostering gatherings based on the topic, but many were billed as *confrontational* panel discussions and/or *contrarian* viewpoints [8]. Over time, however, the topic began to become more seriously inspected, fueled by curiosity as described in [9, 10, 11].

There have been various missteps made throughout the existence of CMMI and its predecessors [12] that contributed to the creation of the Agile Manifesto [13]. Unfortunately, these missteps will continue to be made by people who are not appropriately qualified to be using, appraising, or teaching CMMI (more on that later). Nonetheless, making progress—in a world that accepts structured, deliberate, and persistent improvements together with empowered teams, Lean processes, experimentation, and involved customers—requires that all parties:

- Understand the foundation and intent of both CMMI and Agile.
- Implement CMMI and Agile goals, values, and practices in synergistic ways.

Both Agile and CMMI have been shown to benefit project and organizational performance. To gain the maximum results of a combined approach, appropriate expectations from both must be set. Before this can be done, we must also incorporate particular context of CMMI and Agile.

> *"CMMI focuses on practices and artifacts of cultures of process excellence without addressing the underlying enablers of this culture."*

CMMI Isn't for Everyone—It's for Experts

Whether intentional or not, the body of work that is the CMMI and accompanying products and services (appraisals, training, etc.) in large part targets to an audience of subject-matter experts (SMEs): people with knowledge, training, skills, and a foundation of process improvement theory and practice. There is a tacit assumption that they have broad, practical, applied improvement experience in their particular domain of work and are *professionals* in process improvement. For process improvement SMEs, CMMI is abstract enough, but for people without *a priori* background, experience, and education in process improvement, CMMI is hard to use and lacks sufficient context and background on fundamental process improvement. CMMI, perhaps rightly so, doesn't regress to explain the foundation from which it emerged [1].

CMMI focuses on practices and artifacts of cultures of process excellence without addressing the underlying enablers of this culture. In fact, the term *culture* in any form appears only sparingly in CMMI: twice in reference to choosing a *model representation* (which only really matters when pursuing appraisals, not improvements); once buried in an example within a subpractice of the Organizational Innovation and Deployment process area (which is at Maturity Level 5); once in an example in the introductory notes to the Process and Product Quality Assurance process area; and once in the glossary definition of *institutionalization*. As well, there is no discussion of how to attain a culture conducive to process improvement or what its attributes are.

Although built on decades of process improvement practice—from Deming, Juran, and Crosby to Ohno, Shingo, and the work at Toyota[3]—CMMI doesn't mention these thought-leaders, nor does CMMI directly explain that their work in creating and fostering high-performance organizations is both context to and reflected in CMMI. While not explicitly prerequisites to using CMMI, it stands to reason that having the basic knowledge, technique, and application of process improvement makes success with using CMMI more achievable. The audience for CMMI is assumed to have a working knowledge of and background in basic process improvement.

Just as there are no actual processes in CMMI—and none of the process

areas have enough content to actually create fully functioning processes for actually developing real products and services—CMMI similarly doesn't include the principles and practices of basic process engineering and design. Again, this entry-level content isn't something one would expect to see in an advanced work on process improvement; nonetheless, when people start with CMMI without this context, their failure is predictable.

The contents of the People Capability Maturity Model [14] aren't part of CMMI either. Organizations believe they can achieve high value in CMMI implementations by focusing exclusively on CMMI while ignoring workforce competency and empowerment facets of a process improvement culture (which happen to be among the Agile principles). The attributes and capabilities of high-performing, strongly process-oriented, and well-integrated cultures of improvement are captured in [15].

Agile Limitations (That CMMI Can Mitigate)

Agile values, principles, and practices have been demonstrably beneficial to many organizations. The story is not all rosy, however, and there are plenty of examples of Agile failing to achieve desired outcomes just as there are failures of CMMI to achieve desired outcomes[4]. Failures with Agile often have similar causes to those in CMMI: failing to account for context, background, and culture, and implementing incomplete components.

However, some Agile failures can be mitigated with the practices and constructs in CMMI. Agile values, principles, and practices are mostly oriented at the team and project level and their premises rely heavily on individuals and those currently and immediately involved in a particular effort. CMMI goals and practices assume an organization wants its processes propagated widely and over distances of time and/or space. Therefore, CMMI provides an organizational-level infrastructure as well as mechanisms to:

- Preserve information and knowledge over time.
- Structure and provide criteria for decision-making.
- Strengthen and normalize risk management.
- Methodically apply technical approaches.
- Specifically focus on process improvements.
- Specify engineering practices. (CMMI includes several engineering best practices, but assumes users know the basics).

Additionally, CMMI includes practices for the normalization of processes at the organizational and project levels that are, upon closer inspection, attributes of an organizational culture of process performance. These *generic practices* are frequently overlooked by organizations using Agile approaches since the focus of Agile values is too often applied in a short-term and target effort view, frequently failing to take into account the value of the *things on the right* [2, 13].

> *"Failures with Agile often have similar causes to those in CMMI: failing to account for context, background, and culture, and implementing incomplete components."*

Another area where Agile's content clearly has no material is in the area of quantitative process performance. Some might argue that there is no point to pursuing quantitative optimization since the very notion assumes a process is performed with enough frequency and regularity that pursuit of a quantitative model is value-added.

This is a valid argument, especially for: projects that never have anything in common with other projects; projects that are short-term and/or require minimal effort; organizations that band and disband in an ad-hoc fashion as a function of project scope and client need; and organizations that don't understand that quantitative performance need not be onerous [16]. However, for projects involving more than one or two people, that take months, that have teams whose resource pool (of a dozen or so) is stable and reusable, and for organizations that approach development valuing certain aspects of performance predictability, process optimization is neither out-of-reach nor pointless. In fact, processes can be described, stabilized, normalized, capable, baselined, and optimized with just a few iterations or sprints.

With so much of Agile's content drawing from Lean sources, it is conspicuously lacking in quantitative techniques that are so much of a staple in those very same practices. Concepts such as TQM, Six Sigma, and the Toyota Production System [17, 18, 19]—from which several highly valued Agile principles are based—are all more than principle and culture alone: They are deeply quantitative and steeped in detailed process definitions and standards. Yes, the culture preceded the process definition and statistics, but the process definition and statistics are also a reflection of the culture as well as a facilitator of the relentless pursuit of customer delight. Many practitioners of Agile values and principles stop well short of quantitative techniques and process definitions, thereby making an unwitting shortfall in their own pursuits of excellence.

This is another role played by misinterpretation of CMMI. *Processes* (in the large) are not quantitatively characterized, but rather the focus is on *subprocesses* [20]. Measures are taken at specific points, not at all process junctions, and not throughout a process' use, but where it makes business sense and adds organizational value. With this in mind, even Agile organizations will find that there are many activities they perform with regularity even when projects, teams, and customers are different. Some examples are:

- Refactoring.
- Continuous integration.
- Test-driven development.
- Sprint/iteration planning.
- Planning Poker estimation.
- Pair programming.

Each of these practices are not total processes; they are made-up of subprocesses which can be measured for duration, defects, effort, instances (counts), and so forth. For certain applications within certain projects, there may be benefits to normalizing and then quantifying how some of the underlying subprocesses in these practices perform. Dismissing quantification out-of-hand is nowhere to be found in the Agile values or practices.

Agile Teachings (That CMMI Can Benefit From)

Simplicity. Putting the *basics* back into the improvements.

Among the several catalysts for the

Agile movement were troubling fads in the software development world regarding processes, tools, and methodologies. In particular, these fads were fast approaching the status of trends, pulling the development world into a *vicious cycle* of under-performing project results—despite the ever-increasing presence of concepts ostensibly created to bring about success.

The emergence of the Agile movement did the software development world (and the world of process improvement, in general) a great service by reminding us of the fact that individuals and interactions, working products, customer collaboration, and responding to change do matter more than processes and tools, unnecessary documentation, contract negotiation, and continuing to follow an obsolete plan [16]. Specifically, the things that matter most to the customer are the things the customer perceives, experiences, and pays for. In other words, the things that matter to an organization are facilitated by delivering on things that matter to the customer. Therefore, customer needs are (and always have been) a higher priority to the viability of any business transaction than the *unseen machinations* that enable the organization to deliver against those customer needs.

Is this to say that the unseen machinations are entirely unimportant? No. For the long-term viability of the business, the underlying processes that make the business operate are critical. However, these are less important to the customer than is meeting their immediate expectations. A simple way to reiterate [2] in a process improvement-oriented way is to say *all efforts must align with the needs of the business to satisfy the customer*.

Another practice found among *Agilistas* is that of experimentation: a time-honored process improvement technique long ago lost among the level-manic set. Not just experimenting with solutions in the *fail early and often* Agile sense, but experimenting with processes, organizations, data collection, and tracking techniques. It's not uncommon to hear "... we don't want to try that because it might kill our level."

Mired in the unnecessary bureaucracies of formal process groups, the idea of experimentation with processes has been nearly erased from the process improvement tool kit. In CMMI, the term *experiment* appears exactly three times, not one of which is connected to basic process development. The closest reference to the term is found in the Causal Analysis and Resolution process area (which most organizations don't look at until pursuing Maturity Level 5) as a suggestion within a subpractice related to implementing "action proposals" [21].

Again, this is not a dig at CMMI—it is pointing out the extent of context and knowledge assumed to exist among CMMI users. If any criticism of CMMI in this regard is warranted, it is that the model front-matter does not highlight these assumptions. It leaves hapless project members without a map of steps and tools to navigate by, drowning in the gravity well of process improvement intricacies. The front-matter states:

> The audience for this model includes anyone interested in process improvement in a development and maintenance environment. Whether you are familiar with the concept of capability maturity models or whether you are seeking information to get started on your improvement efforts, this document will be useful to you.
>
> This model is also intended for people who want to use an appraisal to see where they are, those who already know what they want to improve, and those who are just getting started and want to develop a general understanding of the CMMI for Development constellation. [22]

"A danger posed to practitioners of both CMMI and Agile is the incorrect assumption that using either is a substitute or an excuse for ignoring the discipline inherent in actual engineering activities ..."

Without any mention of assumed capabilities, experience, training, aptitudes, knowledge, or skills, CMMI is foisted by level-hungry management on unsuspecting users. Instead, readers most likely expected sufficient content in the model explaining not only what needs to be done, but guiding them in even the most macro-level best practices during implementation. Psychological change management skills and laying the groundwork for the right culture are critical to implementing CMMI, yet are absent from the text. Organizations left without the appropriate understanding are presumed to use CMMI not knowing they lack the wherewithal to get anywhere.

But thanks to Agile's values, principles, and practices and their simple portrayal of the basics of process design and use, organizations can adopt Agile ideas—and with them implement CMMI—without first becoming masters of process improvement. Agile can prevent the bloating of processes, ensure that processes add value, involve the necessary stakeholders, encourage experimentation, and replace level mania with results.

The Perfect Marriage?

Neither CMMI nor Agile include content that replaces thorough engineering practices. A danger posed to practitioners of both CMMI and Agile is the incorrect assumption that using either is a substitute or an excuse for ignoring the discipline inherent in actual engineering activities (or appropriate activities for acquisition and/or services).

For example, an organization that needs to be taught how to analyze requirements to ensure they are necessary and sufficient, or that believes a design can be fully illuminated from conversation alone (and doesn't need description or revisiting), isn't doing engineering and is probably not ready for CMMI. Furthermore, regardless of whether or not such an organization is using CMMI or whether or not they are implementing Agile practices, it isn't actually *developing* in the engineering sense. Such organizations may be programming or coding, but they are not developing. Not performing engineering is nothing more and nothing less than not performing engineering. Having said that, not every project actually requires engineering to be done at the project level. Some projects are merely carrying out the final touches on works that have already been engineered long ago and/or by another organization, leaving the more mechanical work to others. Sadly, it is also too often that engineer-

ing is ignored when it very much ought to be done. This may point to other, deeper issues with software development in general, as raised in [23, 24, 25, 26].

Simply put, when actual engineering is being performed, it produces the appropriate artifacts. Neither CMMI nor Agile can supply enough content to cause (or make necessary) actual engineering practices.

The following examples describe a few ways in which CMMI and Agile were able to work synergistically during development activities.

In one case, Agile practices (Scrum) were already in place and operating effectively. A customer of the organization required a CMMI Maturity Level 2 rating to be compliant with their contract. Although the manager of software development believed there would be a benefit to implementing the practices all the way through Maturity Level 3, executive management preferred the lower expense and faster results of pursuing a Maturity Level 2 rating. Nonetheless, in roughly nine months, the software team of approximately a dozen cross-trained developers and a few specialists had satisfied the Maturity Level 2 goals, also (without explicitly trying) satisfying the goals of at least two or three additional process areas. Their approach included:

- Using Scrum to manage the process deployment and improvement effort and operating the process engineering group as a Scrum team.
- Using measures and metrics that were both easy to obtain and relevant indicators of process, project, organizational, and product performance.
- Integrating CMMI practices into their workflow, including expanding Scrum practices to account for all process activities (regardless of whether they were tied to CMMI or not).
- Leveraging Scrum's product and sprint backlogs, daily stand-up meetings, and end-of-iteration retrospectives for conducting activities that improve their processes.
- Creating a simple developer handbook that explains how product development took place and points its users to all the process assets necessary to do their jobs.
- Creating templates and checklists for the routine aspects of product development and project governance.
- Adding discipline and standards to areas of work that had been allowed to be performed *freestyle*.
- Rotating personnel on and off of the process group to broaden the experience base and keep interest fresh.

None of these attributes of the client's approach are necessarily unique to Scrum/Agile or CMMI; however, the key factor in the client's easy success was that they began with and applied Agile values and practices to implement CMMI, and learned through CMMI the benefits of having certain practices of theirs be more under their control.

This resulted in all projects using a single set of broad practices and measures, increasing the predictability of project activities and the ability to demonstrate progress in process effectiveness. The teams also worked more steadily with fewer disruptions by building specific *touch points* into their workflow. The team also learned that process artifacts were less valuable and more disruptive when not fully designed. Though the discipline that some process activities afforded was beneficial, the team learned that taking the *easy way out* actually proved to be more disruptive than had they created more appropriate tools.

In another case, an organization had been using many practices from Extreme Programming for some time and had experienced dramatic improvements from their previous approach. As a result, they found themselves in the most well-suited position to be leveraged for pursuing Maturity Levels 4 and 5. Rather than burdening themselves with traditional measurement techniques and objectives, the development leader was able to identify several natural *measurement points* throughout their workflow. Such natural measurement points fell into a few general areas:

- Between physical steps in the workflow.
- Wherever automation is able to collect data.
- Whenever data is being entered into or manipulated in a tool.
- During (or subsequent to) refactoring activities.
- At the beginning of iterations where analysis, estimates, and task allocation is performed.
- At the end of iterations (and at releases) where much of the progress-to-date is being reviewed.

One interesting decision was to not attempt certain metrics from the activi-

ties of pair programming. This decision was reached when the impact of the measurement effort was found to exceed the value of the measures. The ability to describe the processes affecting the measures was also determined to be too complex and therefore onerous to control. Despite enormous pressure (and opportunity) to gain valuable data to facilitate high-maturity behaviors, a business decision was made regarding which processes would add value by being optimized (after determining that it would not add business value to attempt to optimize pair programming).

Although this organization has yet to attain the necessary performance and depth of measures to approach Maturity Levels 4 and 5, the lessons they learned and applied from both Agile and CMMI are moving them towards becoming a more highly performing Agile team. One lesson that has been implemented was the realization that greater discipline and finer granularity in backlog management leads to more accurate estimates of task effort. Another lesson was that the intent of CMMI practices may already be accomplished by existing activities when such activities are viewed in broader engineering terms in addition to their more common Agile terms.

Conclusion

CMMI can't be everything to all users. Some users will work with CMMI from the perspective of already being process improvement experts and some will be novices. Regardless, it is easy to take wrong turns with CMMI. However CMMI, in the right hands, can facilitate an evolutionary path towards optimization.

Agile helps improve many operational and transactional activities but wasn't intended to provide higher levels of organizational constructs to facilitate long-term process evolution. Nonetheless, Agile can jump-start effective process design and deployment, and foster a culture of process excellence through its core values in Lean and cooperative processes.

Together, CMMI and Agile can operate synergistically to enhance the other's performance, speed to deployment, and acculturation. Organizations would be well-advised to set aside their prior perceptions of CMMI and Agile compatibility and embrace both their mutually beneficial and shared vision: delivering a high-quality product to the customer on time.◆

References

1. Konrad, Michael D. Personal interview. 2009.
2. Beck, Kent, et al. "Manifesto for Agile Software Development." Feb. 2001 <www.agilemanifesto.org>.
3. Glazer, Hillel. "Dispelling the Process Myth." CROSSTALK Nov. 2001.
4. Paulk, Mark C. "Extreme Programming From a CMM Perspective." *IEEE Software* 18.6 (Nov./Dec. 2001): 19-26 <ftp.sei.cmu.edu/pub/documents/articles/pdf/xp-from-a-cmm-perspective.pdf>.
5. Paulk, Mark C., et al. *Capability Maturity Model for Software*. Version 1.1. SEI, Carnegie Mellon University. Technical Report CMU/SEI-93-TR-024 ESC-TR-93-177. Feb. 1993 <ftp.sei.cmu.edu/pub/documents/93.reports/pdf/tr24.93.pdf>.
6. Paulk, Mark C., et al. *The Capability Maturity Model: Guidelines for Improving the Software Process*. Boston: Addison-Wesley Professional, 1995.
7. Boehm, Barry, and Richard Turner. *Balancing Agility and Discipline: A Guide for the Perplexed*. Boston: Addison-Wesley Professional, 2003.
8. Dalton, Jeffrey. Personal interview. 2007.
9. Glazer, Hillel. "Keys to Making Agile and CMMI Compatible" APLN – Maryland, Chapter Meeting. Audio Podcast and Slides. 15 May 2007 <http://apln.agilemaryland.org/moin/Meeting2007May>.
10. Cook, Linda M. Personal interview. 2007.
11. Dinwiddie, George. Personal interview. 2007.
12. Glazer, Hillel, et al. *CMMI or Agile: Why Not Embrace Both!* SEI, Carnegie Mellon University. Technical Note CMU/SEI-2008-TN-003. Nov. 2008 <www.sei.cmu.edu/reports/08tn003.pdf>.
13. Cockburn, Alistair. Personal interview. 2009.
14. Curtis, Bill, Bill Hefley, and Sally Miller. *People Capability Maturity Model, Version 2.0, Second Edition*. SEI, Carnegie Mellon University. Technical Report CMU/SEI-2009-TR-003 ESC-TR-2009-003. July 2009 <www.sei.cmu.edu/reports/09tr003.pdf>.
15. Spear, Steven J. *Chasing the Rabbit: How Market Leaders Outdistance the Competition and How Great Companies Can Catch Up and Win*. McGraw Hill, 2008.
16. Anderson, David. *Measurement and Analysis in Agile Methods*. Proc. of the 21st Annual SEPG North America Conference. San Jose, CA. 23-26 Mar. 2009.
17. Ohno, Taiichi. *Toyota Production System: Beyond Large-Scale Production*. New York: Productivity Press, 1995.
18. Shingo, Shigeo. *A Study of the Toyota Production System from an Industrial Engineering Viewpoint (Produce What Is Needed, When It's Needed)*. Trans. by Dillon, Andrew P. New York: Productivity Press, 1989.
19. Spear, Steven, and Bowen, H. Kent. "Decoding the DNA of the Toyota Production System." *Harvard Business Review*. Sept.-Oct. 1999 <http://twi-institute.com/pdfs/article_DecodingToyotaProductionSystem.pdf>.
20. CMMI Product Team. *CMMI for Development, Vers. 1.2*. SEI, Carnegie Mellon University. Technical Report CMU/SEI-2006-TR-008. "Organizational Process Performance" section: 261-274. Aug. 2006 <www.sei.cmu.edu/reports/06tr008.pdf>.
21. CMMI Product Team. *CMMI for Development, Vers. 1.2*. "Causal Analysis and Resolution – A Support Process Area at Maturity Level 5" section: 104-106.
22. CMMI Product Team. *CMMI for Development, Version 1.2*. "Preface – Audience" section: iii.
23. Adams, William. With response by Grady Booch. "The Relevance of Architecture." *IEEE Software* 24.5 (Sept./Oct. 2007): 8-9 <http://www2.computer.org/portal/web/csdl/abs/html/mags/so/2007/05/s5008.htm>.
24. Alleman, Glen B. "Bridge Building and Software Development." Weblog post. *Herding Cats*. 16 June 2009 <http://herdingcats.typepad.com/my_weblog/2009/06/bridge-building-and-software-development.html>.
25. "The Role of Software Architect and Software Architecture." *Open Frame Technologies*. 8 Oct. 2009 <www.openframetechnologies.com/Vision1.htm>.
26. Spolsky, Joel. "The Perils of Java Schools." *Joel on Software*. 29 Dec. 2005 <www.joelonsoftware.com/articles/ThePerilsofJavaSchools.html>.

Notes

1. Except where noted, all references to CMMI are to CMMI-DEV v1.2, the model. (CMU/SEI-2006-TR-008)
2. See, for example, discussions at "XP and the CMM" <http://c2.com/cgi/wiki?XpAndTheCmm>; Ronald E. Jefferies' "Extreme Programming and the Capability Maturity Model" <www.

xprogramming.com/xpmag/xp_and_cmm.htm>; and "XP and CMM – Agile and Other Processes Forum at JavaRanch" <www.coderanch.com/t/130051/Agile-Other-Processes/XP-CMM>.

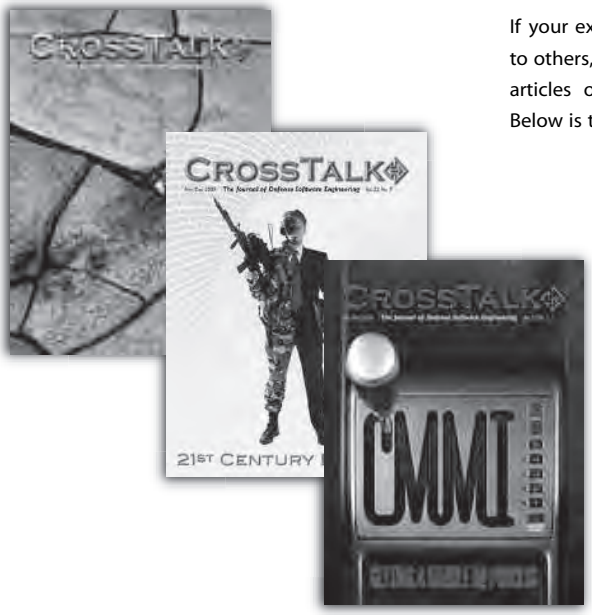
3. W. Edwards Deming, known for "Out of the Crisis," and "The New Economics" (widely credited with Total Quality Management); Joseph M. Juran, known for his development of quality planning, quality control, and quality improvement; Philip B. Crosby, known for "Quality Is Free"; to Shigeo Shingo and Taiichi Ohno, for their work with the Toyota Production System, a precursor to Lean manufacturing; and Sakichi Toyoda, known for developing the "5 Whys."
4. My recent informal survey data from several Internet sources where both Agile and CMMI are discussed revealed that most conflicts are perceived to exist from misapplications of Agile and/or CMMI, not from specific content in either.

About the Author

Hillel Glazer has been working in process improvement since 1991. He is one of the few CMMI High Maturity Lead Appraisers and CMMI Instructors working with Agile teams. Glazer serves as an SEI visiting scientist, contributing to the services constellation while also providing Agile-oriented input to CMMI V1.3. He is the lead author on the SEI's first-ever official publication addressing Agile development. His company, Entinex, focuses on bringing Lean and Agile values back into the formalized process improvement to create advanced states of process and performance excellence.

300 E. Lombard ST # 840
Baltimore, MD 21202
Phone: (410) 814-7513
E-mail: hillel@entinex.com

CALL FOR ARTICLES



If your experience or research has produced information that could be useful to others, CROSSTALK can get the word out. We are specifically looking for articles on software-related topics to supplement upcoming theme issues. Below is the submittal schedule for three areas of emphasis we are looking for:

Catching Up With PSP/TSP
July/August 2010
Submission Deadline: February 12, 2010

Software Assurance Safeguarding
September/October 2010
Submission Deadline: April 9, 2010

Architecture Today
November/December 2010
Submission Deadline: June 11, 2010

Please follow the Author Guidelines for CROSSTALK, available on the Internet at <www.stsc.hill.af.mil/crosstalk>. We accept article submissions on software-related topics at any time, along with Letters to the Editor and BACKTALK. We also provide a link to each monthly theme, giving greater detail on the types of articles we're looking for at <www.stsc.hill.af.mil/crosstalk/theme.html>. CROSSTALK is a peer-reviewed journal.

Article Index
Volume 22

TOPIC	ARTICLE TITLE	AUTHOR(S)	ISSUE	PAGE
21st Century Defense	The Combat-Wireless Health Monitoring System	MAJ Phillip G. Burns	7	4
	Cyberterrorism: The Threat of Virtual Warfare	Summer Olmstead, Dr. Ambareen Siraj	7	16
	PKI: The DoD's Critical Supporting Infrastructure for Information Assurance	Susan Chandler, Jerrod Loyless	7	10
Cost and Schedule Estimation	Understanding Software Project Estimates	Katherine Baxter	3	27
Defect Detection	Defect Detection by Developers	D.T.V. Ramakrishna Rao	3	8
Engineering for Production	Experiences With Software Product Line Development	Dr. Paul Jensen	1	11
	Production Planning for a Software Product Line	Dr. Gary J. Chastek, Linda M. Northrop, Dr. John D. McGregor	1	6
	Software Product Management	Dr. Christof Ebert	1	15
Information Assurance	A Model to Quantify the Return on Information Assurance	Ron Greenfield, Dr. Charley Tichenor	2	18
Language	Enforcing Static Program Properties in Safety-Critical Java Software Components	Dr. Kelvin Nilsen	2	24
Lean Principles	Lean Enablers for Systems Engineering	Dr. Bohdan Oppenheim	5	4
Measurement	Software Project and Process Measurement	Dr. Christof Ebert	5	22
Modeling and Simulation	A Framework for Systems Engineering Development of Complex Systems	Dr. Karl L. Brunson, Dr. Jeffrey Beach Dr. Thomas A. Mazzuchi, Dr. Shahram Sarkani	5	7
	Requirement Modeling for the C-5 Modernization Program	Steven D. Allen, Mark B. Hall, Verlin Kelly, Mark D. Mansfield, Dr. Mark R. Blackburn	2	9
Open Forum	Preparing for an Internal Assessment Interview	Jim O'Brien	7	26
	Resistance as a Learning Opportunity	David P. Quinn	5	32
Programming Tools	Static Analyzers in Software Engineering	Dr. Paul E. Black	3	16
Quality	Management's Inspection Responsibilities and Tools for Success	Roger Stewart, Lew Priven	3	18
Rapid Prototyping	Evolutionary Capabilities Developed and Fielded in Nine Months	Portia Crowe, Dr. Robert Cloutier	4	15
	Using WYSIWYG GUI Tools With UML	Ilya Lipkin, Martin Guldahl	4	4
Requirements	Why Software Requirements Traceability Remains a Challenge	Andrew Kannenberg, Dr. Hossein Saiedian	5	14
Resilient Software	Considering Software Protection for Embedded Systems	Dr. Yong C. Kim, Lt. Col. J. Todd McDonald, Ph.D.	6	4
	Investing in Software Resiliency	Dr. C. Warren Axelrod	6	20
	Making Security Measurable and Manageable	Robert A. Martin	6	26
	Meeting the Challenge of Assuring Resiliency Under Stress	Don O'Neill	6	33
	Resilient Mixed-Criticality Systems	Dr. Lui Sha	6	9
	Software Survivability: Where Safety and Security Converge	Karen Mercedes Goertzel	6	15

TOPIC	ARTICLE TITLE	AUTHOR(S)	ISSUE	PAGE
Service-Oriented Architecture (SOA)	A Perspective on Emerging Industry SOA Best Practices	Larry Pizette, Salim Semy, Geoffrey Raines, Steve Foote	5	29
	Certification and Accreditation of SOA Implementations: Programmatic Rules for the DoD	Anthony David Scott, Michael Malloy, Peter Clay, Mark Masone	7	19
	Leveraging Federal IT Investment With Service-Oriented Architecture	Geoffrey Raines	2	4
Software Acquisition	Defense Acquisition Performance: Could Some Agility Help?	Paul E. McMahon	2	14
Software Assurance	Software Assurance Practice at Ford: A Case Study	Dr. Nancy R. Mead, Dr. Dan Shoemaker, Jeffrey A. Ingalsbe	3	4
Software Development	Three Encouraging Developments in Software Management	Esther Derby	1	28
Software Measurement	Measuring Maintenance Activities Within Development Projects	Lori Holmes, Roger Heller	4	25
Software Process Improvement	Software Process Improvement Implementation: Avoiding Critical Barriers	Dr. Mahmood Niazi	1	24
	"Spending" Efficiency to Go Faster	Dr. Alistair Cockburn	1	20
Software Safety and Security	Software Safety for Model-Driven Development	Timothy J. Trapp, Donald S. Hanline II, Howard D. Kuettner, Jr., William A. Christian	4	9
Software Sizing	The Evolution of Software Size: A Search for Value	Arlene F. Minkiewicz	3	23
System of Systems (SoS)	From Substandard to Successful Software	Martin Allen	4	29
	A Uniform Approach for System of Systems Architecture Evaluation	Michael Gagliardi, William G. Wood, John Klein, John Morley	3	12
Team Software Process	A Distributed Multi-Company Software Project	Dr. William R. Nichols, Anita D. Carleton, Watts S. Humphrey, James W. Over	4	20

Monthly Columns

ISSUE	COLUMN TITLE	AUTHOR
Issue 1: January Engineering for Production	Sponsor: Engineering for Production BackTalk: The Value of Toys	David Curry Dr. Linda R. Wilbanks
Issue 2: February Software and Systems Integration	Sponsor: Software and Systems Integration: A Historical Perspective BackTalk: Two, Four, Six, Eight! Software and Systems–Integrate!	Bruce Amato Dr. David A. Cook
Issue 3: March/April Reinforcing Good Practices	Publisher: Reinforcing Good Practices BackTalk: Real Science ... Fiction	Kasey Thompson Ranse Parker
Issue 4: May/June Rapid and Reliable Development	Publisher: Needing It "Yesterday" BackTalk: Alan Smithee: Where Are You When I Need You? (or, "A Software Engineer Goes to the Movies")	Kasey Thompson Dr. David A. Cook
Issue 5: July/August Process Replication	Sponsor: Process Replication and Model-Based Process Improvement BackTalk: Raiders of the Lost Art	Jeff Schwalb Gary A. Petersen
Issue 6: September/October Resilient Software	Sponsor: Fortifying Our Cyber Defenses BackTalk: What a Great Ride It Was!	Michael A. Brown Nicole French
Issue 7: November/December 21st Century Defense	Sponsor: When Tomorrow Becomes Today BackTalk: Reality Check or Parody Bit?	Karl Rogers Gary A. Petersen

Back issues of CrossTalk can be found online at <www.stsc.hill.af.mil/crosstalk> and are available for download free of charge. As well, a complete listing of Article Indexes from 1994-2009 can be found at <www.stsc.hill.af.mil/crosstalk/2009/articles.html>.

Software Processes? How *Bohr*ing!

After 12 years in the software industry, I recently returned to teaching. I enjoy it immensely, and especially enjoy teaching the "Intro to Computer Science Principles." I find it refreshing to give students their first taste of computer science. I am always amused by their initial first attempts at programming. Just like watching a newborn calf take its first teetering steps, watching a "new mind" learn to write programs is both rewarding and entertaining.

In my graduate courses, however, I expect a somewhat higher standard. I am pretty strict about expecting my students to follow "good" programming practices—good documentation, readable code, etc. My standards are pretty high, and once the students learn what I expect, they eventually see how clear, easy-to-read code makes debugging and maintenance easier (either that, or they humor me until the end of the semester, after grades have been assigned).

In a recent grad class, I was lecturing on some subtle point of operating systems. I was using Unix as an example, and I had pulled up an example of Unix network code from a handy reference manual. As my class was going over the example, several of the students were shocked by some of the "poor" code used in the operating system. Poorly documented. Bad variable names. Heavens, even global variables! A couple of the students pointed out that I would have quickly given them a grade of zero for writing code like that.

My response to them involved the International Obfuscated C Code Contest (IOCCC). The purpose of this contest is to award to most creatively obfuscated code (where obfuscated means to make things as confusing or difficult to understand as possible). A quick example, in the sidebar, is the code that will generate the entire 12 verses of The 12 Days of Christmas. No misprints—I compiled and checked it to make sure! See <http://en.wikipedia.org/wiki/Obfuscated_code>. At first glance, it looks like the random typing of a drunken cat as it trods across the keyboard.

Why do we celebrate poorly written code? Mostly, of course, for the humor. According to the IOCCC Web site <www1.us.ioccc.org/main.html>, it "shows the importance of programming style, in an ironic way."

In the IOCCC, I am sure that every one of the entries was written by an expert coder. They already know how to write very good code—and this contest is a safe forum poking fun at themselves. If my students wrote code like this? First of all, they would never be able to debug it, so grading would be *much* simpler. And I could always hope for a hammer to spring from a keyboard and crack a knuckle if a student tries to write such code. I teach my students to follow the rules, because it allows me to hold them to a standard. However, an experienced programmer who understands the system can safely break a few rules—as long as their experience allows them to do it safely. There are exceptions to every rule, and an expert knows when it's safe to break a rule. But you have to understand the processes *before* you start breaking them.

One reason I love the CMM/CMMI is the emphasis on "repeatable processes." A long time ago, I had a friend try and teach me how to golf. He said I was a perfect student—I consistently made the same mistakes over and over (and over and ...). It's easy to fix a problem when the problem is repeatable.

When designing software, you need to be able to understand a flawed process—and make it better. In software engineering, I teach about heisenbugs (named after the Heisenberg Uncertainly Principle) and bohrbugs (named after the well-defined Bohr atom model) which are bugs that are respectively difficult to reproduce *and* easy to reproduce. A heisenbug can't be reliably duplicated, so it's very hard to find and fix. A bohrbug, however, can be reliably reproduced, making it somewhat easy (or at least easier) to trace and fix.

A process model allows you to create bohrbugs rather than heisenbugs. While I'm not saying it is perfectly OK to create mistakes, I AM saying that if you want to improve your process, you have to at least screw up in a repeatable, predictable manner. And be willing to improve. And promise to NEVER write code like the 12 Days program.

How Not to Write Code
12days.c

```
#include <stdio.h>
main(t,_,a)char
*a;{return!0<t?t<3?main(-79,-
13,a+main(-87,1-_,
main(-
86,0,a+1)+a)):1,t<_?main(t+1,_,a):3,mai
n(-94,-27+t,a)&&t==2?_<13?
main(2,_+1,"%s %d %d\n"):9:16:t<0?t<-
72?main(_,t,
"@n'+,#'/*{}w+/w#cdnr/+,{}r/*de}+,/*{*+
,/w{%+,/w#q#n+,/#{l,+,/n{n+,/+#n+,/#\
;#q#n+,/+k#;*+,/'r :'d*'3,}{w+K
w'K:'+}e#';dq#'l \
q#'+d'K#!/+k#;q#'r}eKK#}w'r}eKK{nl]'/#;
#q#n')){}#}w'){}{nl]'/+#n';d}rw' i;# \
){nl]!/n{n#'; r{#w'r nc{nl]'/#{l,+'K
{rw' iK{;[{nl]'/w#q#n'wk nw' \
iwk{KK{nl]!/w{%'l##w#' i;
:{nl]'/*{q#'ld;r'}{nlwb!/*de}'c \
;;{nl'-
{}rw]'/+,}##'*}#nc,',#nw]'/+kd'+e}+;#'r
dq#w! nr'/ ') }+}{rl#'{n' ')# \
}'+}##(!!/")
:t<-50?_==*a?putchar(31[a]):main(-
65,_,a+1):main((*a=='/')+t,_,a+1)
:0<t?main(2,2,"%s"):*a=='/'||main(0,mai
n(-61,*a,
"!ek;dc i@bK'(q)-
[w]*%n+r3#l,{}:\nuwloca-O;m
.vpbks,fxntdCeghiry"),a+1);}.vpbks,fxnt
dCeghiry"),a+1);}
```

—David A. Cook, Ph.D.
Stephen F. Austin State University
cookda@sfasu.edu

CrossTalk / 517 SMXS/MXDEA
6022 Fir AVE
BLDG 1238
Hill AFB, UT 84056-5820

PRSRT STD
U.S. POSTAGE PAID
Albuquerque, NM
Permit 737

NAVAIR

CrossTalk thanks the above organizations for providing their support.