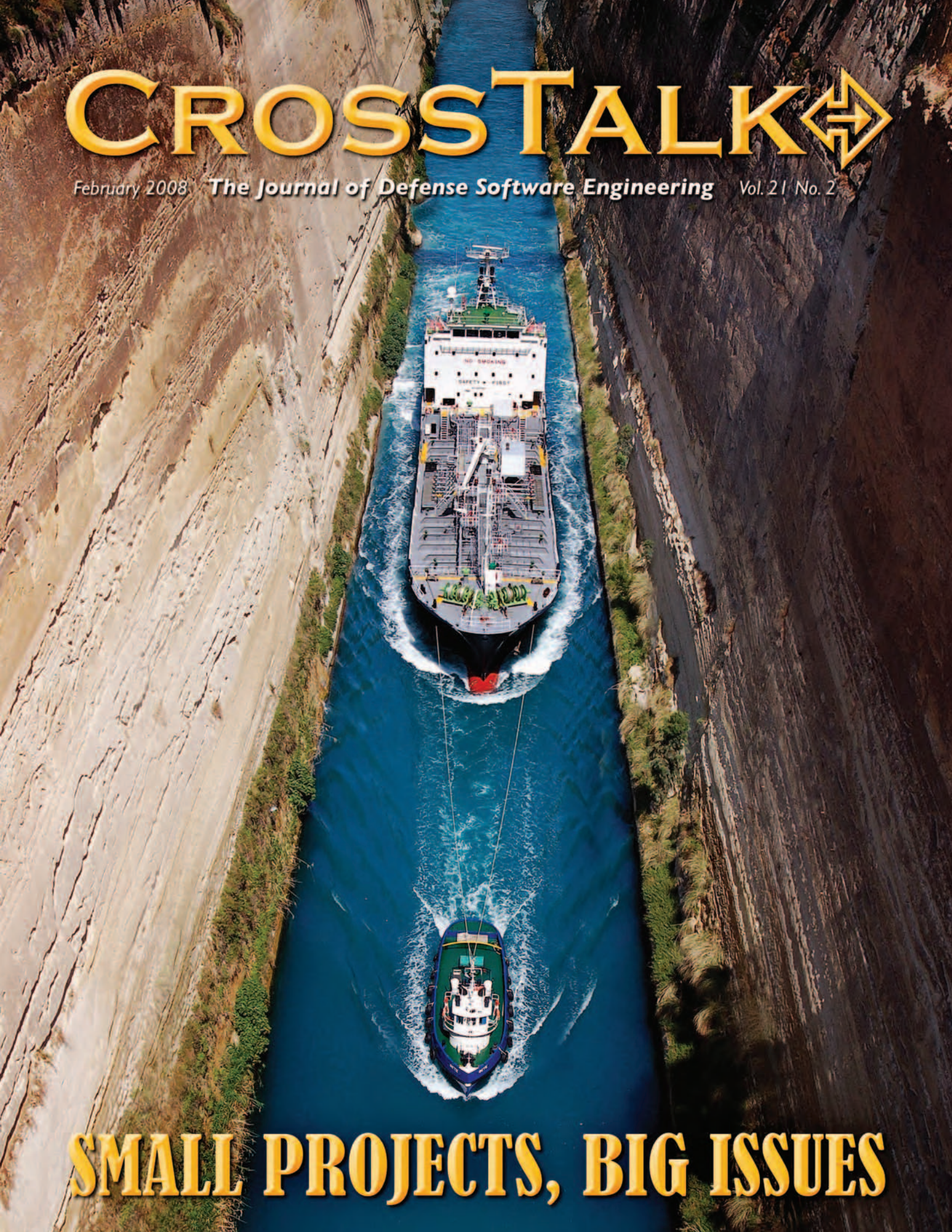
CrossTalk

February 2008 The Journal of Defense Software Engineering Vol. 21 No. 2



SMALL PROJECTS, BIG ISSUES

Small Projects, Big Issues

4 Navigating the Enterprise Forest
Smullin shows that getting integrated into the DoD enterprise and navigating the approval process is a challenge, but with the proper preparation, both are much easier to navigate successfully.
by Fred Smullin

9 Development Practices for Small Software Applications
Jones describes the origins, benefits, and differences of process improvement through the use of CMMI and Agile methods, as well as discusses the benefits of hybrid process improvement models.
by Capers Jones

14 Is CMMI Useful and Usable in Small Settings? One Example
This article presents the motivation, the processes used, and the major results of the CMMI for a Small Business pilot from the perspective of the team that worked on the pilot.
by Sandra Cepeda, Suzanne Garcia, and Jacquelyn Langhout

19 Why Do I Need All That Process? I'm Only a Small Project
This article shows how to fix the issue of having a variety of project sizes but a single set of processes originally built for larger projects.
by Mark Brodnik, Robyn Plouse, and Terry Leip

22 Small Project Survival Among the CMMI Level 5 Big Processes
Jost examines how his organization successfully learned to tailor its CMMI Level 5 processes for small projects.
by Alan C. Jost

Open Forum

27 Field Guide to Provide Step-by-Step Examples for Improving Processes in Small Settings
This article invites contributors to help develop a field guide for process improvement on small projects with how-to guidance, examples, templates, checklists, and other information.
by Caroline Graettinger, Suzanne Garcia, Christian Carmody, M. Lynn Penn, and William Peterson

ON THE COVER

Cover Design by Kent Bingham

Additional art services provided by Janna Jensen

Departments

3 From the Publisher
8 Coming Events
18 Call for Articles
28 Web Sites
29 SSTC 2008
30 CROSSTALK Feedback
31 BACKTALK

CROSSTALK

CO-SPONSORS:

DoD-CIO	*The Honorable John Grimes*
OSD (AT&L)	*Kristen Baldwin*
NAVAIR	*Jeff Schwalb*
76 SMXG	*Kevin Stamey*
309 SMXG	*Norman LeClair*
DHS	*Joe Jarzombek*

STAFF:

MANAGING DIRECTOR	*Brent Baxter*
PUBLISHER	*Elizabeth Starrett*
MANAGING EDITOR	*Kase Johnstun*
ASSOCIATE EDITOR	*Chelene Fortier-Lozancich*
ARTICLE COORDINATOR	*Nicole Kentta*
PHONE	(801) 775-5555
E-MAIL	crosstalk.staff@hill.af.mil
CROSSTALK ONLINE	www.stsc.hill.af.mil/crosstalk

CROSSTALK, The Journal of Defense Software Engineering is co-sponsored by the Department of Defense Chief Information Office (DoD-CIO); 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). DoD-CIO co-sponsor: Assistant Secretary of Defense (Networks and Information Integration). OSD (AT&L) co-sponsor: Software Engineering and System Assurance. USN co-sponsor: Naval Air Systems Command. USAF co-sponsors: Oklahoma City-Air Logistics Center (ALC) 76 Software Maintenance Group (SMXG); and Ogden-ALC 309 SMXG. DHS co-sponsor: National Cyber Security Division of the Office of Infrastructure Protection.

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. 28.

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. Articles published in CROSSTALK remain the property of the authors and may be submitted to other publications.

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.

Good Things Come in Small Packages

"Good things come in small packages." Most people have probably heard this saying from the time they were kids. Small things just don't seem to get the respect they deserve – including small projects. Most of the articles in this month's CrossTalk provide insightful advice for adapting large project processes for small projects. But what about the other side of the story? Small projects have much to offer. Maybe larger projects should consider what they can adapt from small projects. Three areas that large projects may benefit from the lessons learned by smaller projects are Agile techniques, bottom-up estimation, and communication improvements.

There have been efforts to scale Agile practices to large projects. One such effort involves breaking large projects into smaller projects. This might work for the small pieces, but what about when the pieces need to be brought back together? A consistent approach is still needed to pull them together. However, portions of Agile programming can still work on large projects. Pair programming is one such idea: It may seem reasonable that pairing software developers has been shown to decrease errors, and contrary to conventional logic, pairing up programmers also has been shown to increase productivity. The cyclical development of agile programming also scales to large projects and is discussed in depth with the variations of the Spiral Model which is now espoused in the software community. Barry Boehm and his co-authors discuss this thoroughly in back issues of CrossTalk.

Bottom-up software estimation is another technique used in many small projects that may apply to large projects. While it usually is not practical to use bottom-up estimates at the beginning of a large project, bottom-up software estimation is certainly applicable as a sanity check of the estimates as the project progresses, as work is divided into smaller pieces, and as it is assigned to smaller groups and individuals.

And, of course, there is *communication*. A 2002 CrossTalk article discusses new engineers in large companies sitting in their cubicles and accomplishing little because they don't want to make a bother of themselves by asking a lot of questions. Communication tends to be easier on small projects, and someone just sitting will certainly be noticed more readily. Providing tools for communication such as white boards and open space can be implemented in large companies, and the resulting camaraderie has also been shown to improve productivity.

We start this month's issue with Fred Smullin's story of one small project finding its way in the big Department of Defense world in *Navigating The Enterprise Forest*. In *Development Practices for Small Software Applications*, Capers Jones shares his insights, comparing the benefits and drawbacks of the Capability Maturity Model Integration (CMMI) with Agile development for small projects. We next provide a series of articles with expertise on adapting large processes for small projects including *Is CMMI Useful and Usable in Small Settings? One Example* by Sandra Cepeda, Suzanne Garcia, and Jacquelyn Langhout; *Why Do I Need All That Process? I'm Only A Small Project* by Mark Brodnik, Robyn Plouse, and Terry Leip; *Small Project Survival Among the CMMI Level 5 Big Processes* by Alan C. Jost; and *Field Guide to Provide Step-by-Step Examples for Improving Processes in Small Settings* by Caroline Graettinger, Suzanne Garcia, Christian Carmody, M. Lynn Penn, and William Peterson.

On a side note, CrossTalk will be celebrating the 20th anniversary of our first issue in August. We would like to celebrate this milestone with stories of how CrossTalk has helped our readers save time and money, improve processes, salvage projects, and make your jobs easier. We received several success stories with our survey in 2004 – those responses resulted in the continuation of this journal. Your responses now will enable us to thank our co-sponsors for their continued support and strengthen our position as we seek additional co-sponsors.

Large projects tend to fail more frequently than small projects for a variety of reasons. While small projects are figuring out how to benefit from the practices of large projects, large projects can also learn lessons from small projects. Meanwhile, CrossTalk will continue to offer strategies for both.

Elizabeth Starrett

Elizabeth Starrett
Publisher

Navigating the Enterprise Forest

Fred Smullin
Integratable Technologies, LLC

What happens when you take a local depot solution and promote it for worldwide use? You suddenly find yourself facing the challenge of integrating into the Department of Defense (DoD) enterprise with few maps to guide you. I will share my lessons learned as the former chief software architect of the G200 Defense Repair Information Logistics System (DRILS) that started as a local solution and matured into an Air Force solution used both in the .com and .mil domains.

DRILS began as a local maintenance data collection (MDC) system in 2000 to capture the serialized repair of assets. It was commissioned by F-16 supply chain managers (SCMs) to connect them in real time to depot avionics repair activities at Hill Air Force Base. The objective was to collect and analyze vital repair shop information in order to increase the reliability and availability of F-16 avionics as well as decrease repair costs [1]. DRILS most importantly facilitated documentation of individual chips, resistors, and other small parts being replaced within the aircraft avionics components. Other Air Force MDC systems were not able to provide this level of detail, which turned out to be the most important data to SCMs as these parts were the ones that actually failed within the avionics components. The information analyzed from DRILS by the F-16 SCMs under their Falcon Flex program [2] has enabled $133 million in cost avoidance since 2000 and has been projected to achieve approximately $678 million through the aircraft end of life [3].

I will admit that the DRILS stakeholders were naïve to the DoD approval requirements for an Information Technology (IT) system. It would take us more than a year after fielding our initial prototype to navigate our way and be recognized as a legitimate IT system.

The DRILS story is not unlike those of other locally grown data systems in the DoD. An information gap existed within the current mix of maintenance information systems and the depot organization took steps to plug that gap which eventually gave birth to DRILS. You do not have to look far to find information gaps in the DoD. We live in a data rich environment, yet we are information poor; someone somewhere does not have access to the information they need. This basic hunger for dependable information, as well as the lead time and funding required to modify legacy systems, leads to creation of local stovepipe solutions funded and built by the user in that domain.

Tight budgets are impacting the ability to implement local solutions. A 2004 U.S. General Accounting Office (GAO) report found that the DoD requested approximately $19 billion for fiscal year 2004 to operate, maintain, and modernize 2,274 business systems. The report also identified that uncontrolled DoD spending resulted in stovepiped and duplicative systems that included more than 200 inventory and 450 personnel systems being procured and sustained. Very often these stovepipe solutions get thrown over the wall to DoD IT organizations to integrate and sustain within the enterprise [4]. The costs of the integration and sustainment efforts result in priorities getting shifted within existing budgets to accommodate these unplanned requirements.

The National Defense Authorization Act (NDAA) of 2005 was crafted to specifically address this issue [5]. It requires the DoD comptroller to determine that system improvements exceeding $1 million meet the criteria specified in the act. DoD portfolio management (PfM) initiatives have been introduced to comply with the NDAA requirement as well as limit the flow of local solutions that may be stovepiped or duplicative. However, the door is not completely closed to approval of local systems. You can get your local solution approved to operate if you know how to navigate your way through the forest. Based on my experience, if you take the time to address the following questions, you will increase your likelihood of surviving in the enterprise.

What Gaps Are You Filling?

Identify what gaps you are filling in the information food chain. Is there a reason why no one else is providing the information? This is your critical foundation upon which everything else is built. If you are merely churning out the same information that other systems are producing, you will not get far. Focus on those information gaps that prompted you to build the system.

Take those gaps and then describe what is in it for the user community, especially the person doing data entry. Are they getting more out of it than what they put in? If it is not useful to them, you are not going to get your dependable data. Make it worthwhile for them, and you will never be short of dependable data.

Any requirement, given enough time and money, can be integrated into any legacy system. One of the most common reasons a requirement is not integrated is that few organizations have the time and/or money legacy systems request to implement a new requirement. The user community finds it cheaper and faster in the near term to implement their requirements to fill the information gap, and hope to interface it with a legacy system down the road. Unfortunately, this is counterproductive as it just creates more stovepipes.

Try to identify if legacy systems have plans to plug this information gap, and if so, when. If there are plans but they are years out on the horizon, you may get interim authority to operate your gap-filling solution that could evolve into a long-term solution if the implementation is done well.

Develop your own comparison matrix of legacy systems that perform similar functions or may potentially interface to your system. Try to be impartial in your evaluation in order to maximize its credibility. Contact the legacy systems and educate them about what you are doing. Approach them with a partnership offer that assists them with improving the quality of data and information in their system. Document which systems you would interface with if you could and why. Estimate interface costs and return on investment where possible. Interfaces to legacy systems usually provide returns on investment in the areas of duplicate data

entry reduction, minimization of data entry errors, improved data dependability, and near real-time data updates across the enterprise. Keep in mind that there are two costs to any user interface: yours and the legacy system. Your user community may have to pay for both.

In the case of DRILS, we saw that it could potentially interface with the Reliability and Maintainability Information System (REMIS) and the Core Automated Maintenance System (CAMS). Users were frustrated with data entry processes and business rules they perceived to be cumbersome as well as with challenges they encountered trying to analyze historical data. We focused our efforts on streamlining the data entry and data analysis processes for our users. On the depot shop floor, we were able to cut data entry time by 80 percent on average compared to the legacy system. The result was that the volume of depot maintenance data actually increased from the shop floor when compared to the legacy data system. Interfacing with the legacy systems turned out to be the greatest approval challenge. It took nearly six years of meetings, briefings, and requests for funding before the official system interface was allowed to be built with REMIS. A CAMS interface is still actively being pursued at this date.

One lesson learned is that headquarters is generally willing to entertain temporary solutions to initially fill information gaps. Locally developed temporary solutions are usually more agile and less costly than legacy systems to experiment with. Thus, temporary solutions are excellent proving grounds for defining requirements to be incorporated into the legacy system in the long term.

This proved true for DRILS: Our user community encompassed a relatively large portion of the F-16 avionics community but was still small when compared to Air Force-wide MDC legacy systems. Our development and support teams were also much smaller which shortened our decision-making time. The architecture design allowed us to isolate experimental modules from all user communities except those participating. Thus, our cost to implement requested changes for experimental initiatives was significantly smaller and our time to value was also significantly shorter. This made DRILS an ideal system to support MDC experiments such as Air Force Serial Number Tracking initiatives sponsored by high-level champions.

Who Are Your Champions?

A key to successfully implementing any IT system in the DoD is to identify champions within the customer and user base. Champions identified within this community can help advocate the system at the various levels of review and approval. These champions need to be evangelistic because their support will be tested up the chain of approval. They will need to be able to communicate their need and why your solution is the best.

For example, in the case of DRILS, we were fortunate that the product concept and implementation sold itself. Several levels of champions sprung up during the product implementation. The SCMs wanted the repair data from the shop floor and convinced the repair shop supervision to give it a try. Supervision asked their data entry technicians to try the system. They did with some reluctance, but once they started entering data, they became believers.

"A key to successfully implementing any IT system in the DoD is to identify champions within the customer and user base. Champions identified within this community can help advocate the system at the various levels of review and approval."

Technicians in depot repair shops are graded on their production, and the overall organization is graded on its ability to produce quality assets on schedule and on budget. Any impediment to those goals can impact their customers and cost them workload in their competitive environment. Legacy MDC systems used until then had been deemed cumbersome to use by those using it and did not provide any perceivable value to those technicians in meeting quality, budget, and schedule. The DRILS development team lived on the shop floor for nearly four years working hand in hand with the using technicians to refine how the data was collected and displayed. This enabled the system to provide immediate payback to the person entering the data.

This focus on the technician at the point of maintenance enabled them to proactively identify issues that most likely would not have been detected with the legacy systems. A real-world example involved the F-16 multi-function displays whose newly manufactured replacement power supplies that cost $5,000 each were failing within five months of installation. The DRILS design enabled the technician to easily notice the trend, stop installing those parts, and alert the SCM who triggered an investigation with the manufacturer. That investigation eventually led to the identification of a defect in the manufacturing process. Without DRILS, the trend may have gone on for many more months and possibly grounded aircraft due to failed parts clogging the supply chain and consuming financial resources.

Stories such as these enabled long-standing issues to start getting fixed. These success stories gained the attention of the warfighter customer who then wanted the system adapted for their use. This sold the supervisor who in turn sold their Colonel who in turn sold his Brigadier General. The Brigadier General then raised awareness all the way to the Office of the Secretary of Defense. Word started to spread between the weapon systems and Major Commands when warfighters who had used the system moved from unit to unit. It was not long before we had obtained several levels of champions. But our most critical level continued to be the data entry person at the point of maintenance.

How Do You Align With the Mission?

Another key to winning acceptance in the DoD enterprise is to identify how you align with the overall IT mission of your agency and the DoD in general. Obtain copies of headquarters briefings in your domain and examine their road map and the issues they are trying to solve. How do you fit within that road map? If you can show how you fit within that road map, you can gain critical awareness and possibly acceptance at the headquarters level. Part of gaining acceptance is educating them about how you fit with current legacy systems.

Compliance With Standards

The IT industry continues to evolve toward a net-centric world where standards-based computing is pushing out proprietary products in order to facilitate easier integration in heterogeneous envi-

ronments. The DoD continues to gravitate to these standards, although slower than industry, for the same reasons. It is important that you inventory applicable technology standards in your application domain as well as those of the mission you are supporting and remain consistent with those established standards.

DRILS is a Web-based Air Force maintenance data collection system; well-published standards for maintenance data existed in Technical Order 00-20-2 and other publications [6]. We had to remain consistent with those standards at a minimum in order to be able to feed data to the legacy MDC systems in order to allow a much broader Air Force audience to analyze the data. Technology-wise, we intentionally selected well-known commercial off-the-shelf products and kept those products to a minimum in order to avoid integration headaches while remaining consistent with the Global Combat Support System – Air Force requirements.

Can You Participate in a Pathfinder Initiative?

Pathfinder initiatives are very beneficial. Merriam Webster's online dictionary defines pathfinder as, "one that discovers a way; *especially*: one that explores untraversed regions to mark out a new route" [7]. My experience with pathfinder initiatives has involved a charter between a particular community of interest and the headquarters to solve a process or information gap. These pathfinders involve assembling members of the community of interest to review and improve processes and policies. In order to establish a baseline and measure the effect of change, the pathfinder members must collect data. Thus, appropriate data systems are selected as tools to provide the data.

For example, the Air Force decided to initiate a Reliability Pathfinder to study and define the benefits of Item Unique Identification and Automated Identification Technology (AIT) in regards to facilitating serial number tracking within the maintenance processes. The pathfinder team members analyzed the available data systems and chose to use DRILS as the tool with which to collect their data. They performed their analysis on selected B-52 avionics maintenance occurring on the flight line and at the depot. DRILS was used basically as is with a few minor software modifications to facilitate specific data collection and analysis. The result is that the Air Force Reliability Pathfinder has proven very successful. Reports are currently being prepared that have the potential to positively impact the future of serialized asset tracking.

What I learned while participating in three Air Force and two joint Air Force and Army service pathfinder initiatives is that they are useful for unifying a vision. Headquarters depends on field users to define requirements for them. Users want headquarters to make decisions and investments that will improve their work environment, but often do not know how to effectively communicate requirements. I saw disconnects occurring on both sides.

> *"A pathfinder initiative provides an excellent forum ... to collaborate in a closed environment, reach a common understanding, solve longstanding issues, and communicate those solutions to all parties. If you can team with an existing legacy system ... then you have significantly increased your odds of being approved."*

A disconnect may occur in the understanding of the big picture at the user level, while the headquarters may not completely understand the detailed needs of the user. A pathfinder initiative provides an excellent forum for these two groups to collaborate in a closed environment, reach a common understanding, solve longstanding issues, and communicate those solutions to all parties.

To participate in a pathfinder, you need to apply your gap assessment, the backing and breadth of your champions, and your legacy system comparisons to make your case to headquarters of how you can help with a pathfinder effort. If you can team with an existing legacy system to solve the pathfinder needs, then you have significantly increased your odds of being approved.

Pathfinder efforts are as resource-challenged as any other program. Therefore, financial resources to support your efforts will be very limited. However, the exposure and lessons learned from a successful pathfinder effort are significant. Pathfinder progress reports are reviewed at the highest management levels. A successful pathfinder effort will often lead to other pathfinders that increase the exposure of your system as well as your acceptance within the DoD IT community.

Do You Have Portal Capability?

How many user names and passwords do you currently maintain? Do you think users will be willing to add your system to the list as well? I am personally aware of an office that did a Lean study and found they lost 1.5 hours of productivity per day logging in and out of 22 data systems to do their job. This frustrated the workers and decreased their overall job satisfaction.

You can increase your probability of user acceptance by checking to see if there is a portal such as the Air Force Portal or Army Knowledge Online (AKO) that you can integrate with to provide streamlined sign-on capability. Check with your portal for specific requirements. Interfacing with a portal will also demonstrate your ability to integrate within the enterprise, and decision makers will often sway your way when compared to a non-integrated system.

In the case of DRILS, we were able to integrate the application with the Air Force Portal that streamlined sign-on for many of our DoD users. It also allowed us to extend use of the application to selected F-16 DoD repair contractors in the .com world that facilitated increased visibility of F-16 avionics repairs worldwide.

The DRILS Air Force Portal integration went fairly smoothly with only relatively minor edits to our authentication process. We did encounter policy challenges that we felt had to be overcome. We had several hundred users who depended on the application for depot production. If the portal went offline, we risked not collecting valuable data as production would continue, but data capture may not catch up. Thus, we still wanted our users to be able to access the system. The Air Force Portal policy was that our application authentication must be restricted to portal users and deny direct access and login via our non-portal Web address. It took a few e-mails and conference calls as well as a formal waiver

request to gain approval for a hybrid security model that would allow both Air Force Portal and manual authentication. This allowed us to ensure maximum availability to at least our .mil users. Those in the .com world would have to remain dependent on the Air Force Portal availability.

Have You Done Your Paperwork?

I dislike doing paperwork as much as the next person. Unfortunately, paperwork is just part of the territory when it comes to building and fielding a DoD IT system. Recent NDAA legislation leaves you with little choice. You risk incurring stiff financial and judicial penalties if you do not complete your paperwork.

The following questions address the two main documentation areas that should be common across the DoD. Each agency may impose additional requirements. You will need to check with your respective agency for details.

Are You Registered With PfM?

PfM is your required first approval stop for any local or global data system. PfM is the management of selected groupings of investments using integrated strategic planning, integrated architectures, performance measures, risk-management techniques, transition plans, and portfolio investment strategies. The PfM process is driven by a number of legislative acts and DoD directives.

At the root of PfM is the Clinger-Cohen Act of 1996 that requires agencies to use a capital planning and investment control process to provide for selection, management, and evaluation of IT investments [8]. Consequently, the DoD published Directive 8115.01, Information Technology Portfolio Management, to establish policy and assign responsibility for the management of DoD IT investments as portfolios that focus on improving DoD capabilities and mission outcomes [9]. DoD Directive 8000.1, Management of DoD Information Resources and Information Technology, establishes the requirement for a Chief Information Officer (CIO) role in the agencies to manage these portfolios. The CIOs designate portfolio managers to manage their portfolios [10]. Portfolio managers interact with DoD IT system program managers to report the status of their programs.

The DoD Enterprise Information Technology Portfolio Repository (DITPR) is one system used to track portfolios. DITPR was selected by the DoD CIO as the enterprise shared space for IT PfM data for all DoD business IT systems. However, each branch has its own methods of IT registry that feed to DITPR. The Air Force uses the Enterprise Information Technology Data Repository (EITDR), the Navy and Marines use the DITPR-DON (Department of Navy) system, and the Army uses the Army Portfolio Management System (APMS) as their registry. All of these systems are used to record investment review and certification submission information, Federal Information Security Management Act (FISMA) of 2002 assessments, and more [11]. The IT Lean acquisition process and security, interoperability, supportability, sustainability, and usability processes are integrated into these systems as well.

These systems are necessary in order to provide portfolio managers access to information needed to do the following: maximize value of IT investments while minimizing risk, improve communication and alignment between IT and DoD leaders, facilitate team thinking versus individual commands or units, enable more efficient use of assets, reduce the number of redundant projects and eliminate non-value added projects, and support an enterprise IT investment approach.

Portfolio managers depend on IT program managers to keep the portfolio data current for their respective systems in order to fulfill their goals. Participation in PfM is not an option. There are serious consequences for not complying with all of the PfM requirements.

If you are building a new system or expanding the capability of an existing system, you must submit a capability request to your respective portfolio manager to get authorization. This is where you once again tap into your foundational data that describes the gaps you are filling. You may need to arrange a meeting with your portfolio manager to describe why you need the capability and that a similar capability does not exist. The portfolio manager has to weigh a lot of criteria when making a decision to authorize your request and may request additional data to reach their decision. You may even be invited to a *fly off* before a board who is evaluating similar systems within the portfolio.

Do You Meet the Information Assurance Requirements?

One of the first things that a portfolio manager will evaluate is whether you comply with mandatory information assurance (IA) requirements for your system. IA is more important today than it ever has been; information warfare attacks are a reality. FISMA requires each federal agency to develop, document, and implement an agency-wide program to provide information security for the information and information systems that support the operations and assets of the agency [11].

In order to comply with FISMA requirements, the DoD has created the Defense Information Assurance Certification and Accreditation Process (DIACAP) that replaced the Defense Information Technology Security Certification and Accreditation Process. DIACAP assigns, implements, and validates DoDI 8500.2 standardized IA controls and manages IA posture across DoD information systems consistent with FISMA legislative policy as well as DoD regulatory policy found in the 8500 series of directives [12].

You need to ensure that your system remains compliant with the IA requirements identified in the DoD IA 8500 series of directives. If you do not, then you will not be authorized to operate on the DoD network or interface to legacy systems. DoDI 8500.2 assigns IA controls to three Mission Assurance Categories (MAC) and three data sensitivity levels. You will need to evaluate your system and select one MAC and one data sensitivity level appropriate to your system and mission that will determine what your IA control requirements are.

Once you have shown that you comply with the IA control requirements, you must submit a Certification and Accreditation (C&A) package to your Designated Approval Authority for approval using the DIACAP workflow. When your package is approved, an Authority to Operate will be issued that is valid for three years from the date it is issued. DIACAP also requires annual security reviews of the C&A package and those reviews are reported to the portfolio manager through the appropriate portfolio registry such as EITDR, DITPR-DON, or APMS.

Complying with mandatory IA requirements is just one piece of the security puzzle. You need to also ensure the system is programmed defensively using secure coding techniques to ensure that your system and its information are not compromised. Web application security is considered a weak point in an IT security wall and subject to information warfare attack.

The Defense Information Systems

Coming Events

March 3-7
SD West 2008 Software Development Conference and Expo West
Santa Clara, CA
http://sdexpo.com

March 4-5
Warfighter's Vision 2008
Tampa, FL
www.afei.org

March 11-12
2008 Military and Aerospace Electronics Forum
San Diego, CA
http://mtc08.events.pennnet.com/fl//index.cfm

March 16-18
2008 Engineering Research Council Summit, Workshop and Forum
Arlington, VA
www.asee.org/conferences/erc/2008/index.cfm.

March 17-20
2008 SEPG
Tampa, FL
www.sei.cmu.edu/sepg

March 18-20
Sea - Air - Space 2008
Washington, D.C.
www.sasexpo.org/2008

March 24-28
International Testing Certification Super Week
Chicago, IL
www.testinginstitute.com

April 29-May 2
2008 Systems and Software Technology Conference
Las Vegas, NV
www.sstc-online.org

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

Agency has published a Security Technical Implementation Guide titled "Application Security and Development Security." This can be downloaded at <http://iase.disa.mil>.

Conclusion

It is possible to grow a local product into an enterprise system with today's increased PfM and IA requirements. We started DRILS in July of 2000, delivered our rapid prototype in September of 2000, and used evolutionary development from that point forward. During my six years as chief architect, I saw a lot of transformation on how IT systems are certified and supported. I am glad to say that it is becoming less of a paperwork drill now. However, there are still a lot of steps to be checked off. You still have to do your homework and some paperwork to lay your foundation in order to educate your user community, champions, and portfolio managers on why your system should exist.

Align yourself wherever possible with the goals, objectives, and standards of your agency and the DoD in general. Pathfinders are an excellent avenue to prove your alignment, increase your visibility, and gain acceptance at the headquarters level. You can further demonstrate your capability to integrate in the enterprise by facilitating streamlined sign-on to your application through a portal such as the Air Force portal or AKO.

Getting integrated into the DoD enterprise is not only a technology challenge but also a challenge of navigating the approval process. However, with the proper preparation the approval process will be much easier to navigate successfully.◆

References

1. Lindsey, Capt. Greg, and Kevin Berk. "Serialized Maintenance Data Collection Using DRILS." CrossTalk Oct. 2003 <www.stsc.hill.af.mil/CrossTalk/2003/10/0310lindsey.pdf>.
2. Berk, Kevin. "Falcon Flex: Turning Maintenance Information into Air Power." Defense AT&L July-Aug. 2007 <www.dau.mil/pubs/dam/2007_07_08/lebr_ja07.pdf>.
3. Total Quality Systems. "UID/Falcon Flex Transforming Sustainment." Proc. of the U.S. Air Force UID/AIT Conference 2007 <www.dla.mil/j-6/AIT/Conferences/USAF_UID-AIT_Conference/default.aspx>.
4. U.S. GAO. DoD Business Systems Modernization – Billions Continue to Be Invested With Inadequate Management Oversight and Accountability. GAO-04-615. Washington: GAO, 2004.
5. Congress of the United States of America. Ronald W. Reagan National Defense Authorization Act for Fiscal Year 2005. 108th Congress, 2004 <http://thomas.loc.gov/cgi-bin/query/z?c108:H.R.4200.enr:>.
6. U.S. Air Force. "Technical Order 00-20-2, Maintenance Data Documentation." Apr. 2007.
7. Merriam Webster <www.merriamwebster.com/dictionary/ pathfinder>.
8. Congress of the United States of America. Clinger-Cohen Act of 1996. 104th Congress, 1996 <www.cio.gov/Documents/it_management_reform_act_Feb_1996.html>.
9. DoD. "DoDD 8115.01, Information Technology Portfolio Management." Oct. 2005 <www.dtic.mil/whs/directives/corres/pdf/811501p.pdf>.
10. DoD. "DoDD 8000.1, Management of DoD Information Resources and Information Technology." Feb. 2002 <www.js.pentagon.mil/whs/directives/corres/pdf/800001p.pdf>.
11. United States. "Federal Information Security Management Act (FISMA)." 2002 <www.whitehouse.gov/omb/egov/g-4-act.html>.
12. DoD. "DoDD 8500.2, Information Assurance Implementation." Feb. 2003 <www.dtic.mil/whs/directives/corres/pdf/850002p.pdf>.

About the Author

Fred Smullin is founder and president of Integratable Technologies, LLC. He has been involved in developing software for DoD customers over the past 18 years, as well as consulting internationally on commercial software projects. Smullin served as the Chief Software Architect of the G200 DRILS from 2000 to 2006 before launching Integratable Technologies, LLC. His passion is researching how to make enterprise integration easier.

Integratable Technologies, LLC
1436 Legend Hills DR
STE 105
Clearfield, UT 84015
Phone: (801) 779-1035
Fax: (801) 779-1057
E-mail: fsmullin@integratabletech.com

Development Practices for Small Software Applications©

Capers Jones
Software Productivity Research, LLC

Because large software projects are troublesome and often out get out of control, a number of effective development methods have been created to help reduce the problems of large software projects. Two of these methods include the well-known Capability Maturity Model® (CMM®) and the newer CMM Integration℠ (CMMI®) of the Software Engineering Institute (SEI). When these methods are applied to small projects, it is usually necessary to customize them because, in their original form, they are somewhat cumbersome for small projects, although proven effective for large applications. More recently, alternate methods derived from smaller software projects have become popular under the general name of Agile software development. Since the Agile methods originated for fairly small projects, they can be applied without customization and often return very positive results. It is possible to merge Agile concepts with CMM and CMMI concepts, but projects cited in this article did not do so because such mergers are comparatively rare.

Software applications come in a wide range of sizes and types. Each size and type tends to have evolved typical development practices. For example, military projects are usually much more formal and perform many more oversight and control activities than civilian projects. Systems software tends to have more kinds of testing and quality control activities than information systems.

When application sizes are considered, there are very significant differences in the development processes that are most commonly used for large software projects compared to small software projects.

Before discussing size differences, it is best to start by defining what is meant by the terms *large* and *small*. Although there is no agreed-to definition for these terms, the author generally regards *large* applications as being those whose size exceeds 10,000 function points or about 1,000,000 source code statements. The author regards *small* applications as those whose size is at or below 1,000 function points or about 100,000 source code statements.

It is a significant observation that out of 16 software lawsuits where the author served as an expert witness, 15 of them were for applications in the 10,000 function point size range, or larger. The smallest application noted during litigation was 3,000 function points.

Usually software applications of 1,000 function points or below are fairly trouble-free and, therefore, seldom end up in litigation for outright failure, quality problems, cost overruns, or schedule slips. On the other hand, an alarming percentage of applications larger than 10,000 function points exhibit serious quality problems, outright failure, or massive overruns.

Some readers might think that 1,000 function points or 100,000 source code statements are still rather large. However, there are very few commercial software applications or business applications that are smaller than this size range. Software in the range of 100 function points or 10,000 source code statements usually consists of enhancements to larger applications rather than stand-alone applications. Even the smallest commercial off-the-shelf software packages are in the range of 1,000 function points.

> *"When application sizes are considered, there are very significant differences in the development processes that are most commonly used for large software projects compared to small software projects."*

The overall size range of software applications noted by the author runs from a high of about 300,000 function points (30,000,000 source code statements) down to about 0.1 function points (10 source code statements) for a small bug repair.

At the extreme high end are very few massive applications such as enterprise resource planning (ERP) packages and some large defense systems. Operating systems and major application packages such as Microsoft Office are in the range of 100,000 function points or 10,000,000 source code statements. Various components of Microsoft Office such as Excel, Word, PowerPoint, etc., are between 5,000 and 10,000 function points in size.

Origins of the CMM and Agile Development

The SEI was incorporated in 1984 and is located at Carnegie Mellon University in Pittsburgh, Pennsylvania. The SEI was originally funded by the Defense Advanced Research Projects Agency.

Some of the major concerns that led to the creation of the SEI were the very serious and common problems associated with large software projects. One of the early, major, and continuing activities of the SEI was the development of a formal evaluation or assessment schema for evaluating the capabilities of defense contractors, which was termed CMM. Watts Humphrey's book on this topic, "Managing the Software Process" became a bestseller [1].

The initial version of the CMM was published by SEI in 1987, and continued to grow and evolve for about 10 years. Development of the CMM more or less stopped in 1997 and the emphasis switched to the next generation, or the CMMI. The CMMI was initially published in 2002 and has been updated several times. Associated with the CMMI are Watts Humphrey's Team Software Process℠ (TSP℠) and Personal Software Process℠ (PSP℠).

The SEI assessment approach is now

© 2007 by Capers Jones. All rights reserved.
® Capability Maturity Model, CMM and CMMI are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.
℠ CMM Integration, Team Software Process, Personal Software Process, TSP, and PSP are service marks of Carnegie Mellon University.

well documented and well covered in software literature. Indeed, Addison Wesley Longman has an entire series devoted to SEI assessments. The SEI assessment data is collected by means of on-site interviews using both a standard questionnaire and also observations and informal data.

Because of the importance of very large systems to the Department of Defense, the SEI assessment approach originally dealt primarily with the software processes and methodologies used by large companies that produced large systems. The original SEI assessment approach was derived from the best practices used by leading corporations such as IBM and ITT, which employ from 5,000 to more than 50,000 software professionals, and which could safely build systems in excess of 1,000,000 lines of code or 10,000 function points.

Based on the patterns of answers to the SEI assessment questions, the final result of the SEI assessment process places the software organization on one of the levels of a five-point maturity scale. The five plateaus of the SEI maturity levels are shown in Table 1.

It is immediately obvious that the distribution of software organizations is skewed toward the low end of the scale. A similar kind of skew would occur if you were to look at the distribution of candidates selected to enter the Olympics for events such as downhill skiing. Most ordinary citizens could not qualify at all. Very few athletes could make it to the Olympic tryouts, even fewer would represent their countries, and only three athletes from around the world will win medals in each event.

As data is collected, it becomes evident that there is quite a bit of overlap among the various SEI maturity levels. For example, in terms of both quality and productivity, the best software projects from Level 1 organizations can be superior to the worst developed by Level 3 organizations, although the statistical averages of Level 3 are far better than those of Level 1.

There is now fairly solid evidence about the CMM from many studies. When organizations move from CMM Level 1 up to Level 2, 3, 4, and 5, their productivity and quality levels tend to improve based on samples at each level.

As of 2007, the newer CMMI has less empirical data than the older CMM, which is not surprising given its more recent publication date. However, the TSP and PSP methods do have enough data to show that they are successful in improving both quality and productivity at the same time.

When the CMM originated in the 1980s, the *waterfall* method of development was the most common at that time and was implicitly supported by most companies that used the early CMM. However, other methods such as spiral, iterative, etc., were quickly included in the CMM as well. The CMM is neutral as to development methods, but among the author's clients who adopted the CMM, about 80 percent also used the waterfall method.

"What the CMM provided was a solid framework of activities, much better rigor in the areas of quality control and change management, and much better measurement of progress, quality, and productivity than was previously the norm."

What the CMM provided was a solid framework of activities, much better rigor in the areas of quality control and change management, and much better measurement of progress, quality, and productivity than was previously the norm.

Table 1: *Five Levels of the CMM*

SEI Maturity Level	Meaning	Frequency of Occurrence
1 = Initial	Chaotic	75.0%
2 = Repeatable	Marginal	15.0%
3 = Defined	Adequate	8.0%
4 = Managed	Good to excellent	1.5%
5 = Optimizing	State of the art	0.5%

The history of the Agile methods is not as clear as the history of the CMM because the Agile methods are somewhat diverse. However, in 2001 the famous Agile Manifesto was published [2]. This provided the essential principles of Agile development. That being said, there are quite a few Agile variations including eXtreme Programming (XP), Crystal Development, Adaptive Software Development, Feature Driven Development, and several others.

Some of the principle beliefs found in the Agile Manifesto include the following:

- Working software is the goal, not documents.
- Working software is the principle measure of success.
- Close and daily contact between developers and clients is necessary.
- Face-to-face conversation is the best form of communication.
- Small, self-organizing teams give the best results.
- Quality is critical, so testing should be early and continuous.

The Agile methods, the CMM, and the CMMI are all equally concerned about three of the same fundamental problems:

1. Software requirements always change.
2. Fixing software bugs is the most expensive software activity in history.
3. High quality leads to high productivity and short schedules.

However, the Agile method and the CMM/CMMI approach draw apart on two other fundamental problems:

1. Paperwork is the second most expensive software activity in history.
2. Without careful measurements, continuous progress is unlikely.

The Agile method takes a strong stand that paper documents in the form of rigorous requirements and specifications are too slow and cumbersome to be effective. In the Agile view, daily meetings with clients are more effective than written specifications. In the Agile view, daily team meetings or Scrum sessions are the best way of tracking progress, as opposed to written status reports. The CMM and CMMI do not fully endorse this view.

The CMM and CMMI take a strong stand that measurements of quality, productivity, schedules, costs, etc., are a necessary adjunct to process improvement and should be done well. In the view of the CMM and CMMI, it is hard to prove that a methodology is a success or not without data that demonstrates effective

progress. The Agile method does not fully endorse this view. In fact, one of the notable gaps in the Agile approach is any quantitative quality or productivity data that can prove the success of the methods.

Differences in Development Activities Between the Agile and CMM/CMMI Methods

In many industries, building a large product is not the same as building a small product. Consider the differences in specialization and methods required to build a wooden kayak versus building an 80,000-ton cruise ship. A kayak can be constructed by a single individual using only hand tools, but a large, modern cruise ship requires more than 500 workers including specialists such as pipe fitters, electricians, steel workers, welders, painters, interior decorators, air conditioning specialists, and many others.

Software follows a similar pattern: Building a large system in the 10,000 function point range is more or less equivalent to building other large structures such as ships, office buildings, or bridges. Many kinds of specialists are utilized and the development activities are quite extensive compared to smaller applications.

Because the CMM approach was developed in the 1980s when the waterfall method was common, it is not difficult to identify the major activities that are typically performed. For an application of 1,000 function points (approximately 100,000 source code statements), the following 20 normal activities were noted among the author's clients who used either the CMM or CMMI:

1. Requirements.
2. Prototyping.
3. Architecture.
4. Project planning and estimating.
5. Initial design.
6. Detailed design.
7. Design inspections.
8. Coding.
9. Reuse acquisition.
10. Code inspections.
11. Change and configuration control.
12. Software quality assurance.
13. Integration.
14. Test plans.
15. Unit testing.
16. New function testing.
17. Regression testing.
18. Integration testing.
19. Acceptance testing.
20. Project management.

Using the CMM and CMMI, the entire application of 1,000 function points would have the initial requirements gathered and analyzed, the specifications written, and various planning document produced before coding got under way.

By contrast, the Agile method of development would follow a different pattern. Because the Agile goal is to deliver running and usable software to clients as rapidly as possible, the Agile approach would not wait for the entire 1,000 function points to be designed before coding started.

What would be most likely with the Agile methods would be to divide the overall project into four smaller projects, each of about 250 function points in size. In Agile terminology, these smaller segments are termed iterations or sometimes *sprints*.

However, in order to know what the overall general set of features would be, an Agile project would start with *Iteration 0* or a general planning and requirements-gathering session. At this session, the users and developers would scope out the likely architecture of the application and then subdivide it into a number of iterations.

Also, at the end of the project when all of the iterations have been completed, it will be necessary to test the combined iterations at the same time. Therefore, a release phase follows the completion of the various iterations. For the release, some additional documentation may be needed. Also, cost data and quality data need to be consolidated for all of the iterations. A typical Agile development pattern might resemble the following:

- **Iteration 0**
  1. General overall requirements.
  2. Planning.
  3. Sizing and estimating.
  4. Funding.
- **Iterations 1-4**
  1. User requirements for each iteration.
  2. Test planning for each iteration.
  3. Testing case development for each iteration.
  4. Coding.
  5. Testing.
  6. Scrum sessions.
  7. Iteration documentation.
  8. Iteration cost accumulation.
  9. Iteration quality data.
- **Release**
  1. Integration of all iterations.
  2. Final testing of all iterations.

Successful Hybrid Approaches to Software Development

One of the major trends in the industry since about 1997 has been to couple the most effective portions of various software development methodologies to create hybrid approaches. These hybrid methods are often successful and often achieve higher quality and productivity rates than the original *pure* methods. As of 2007 some interesting examples of the hybrid approach include, in alphabetical order:

- Agile joined with object-oriented development (OOD).
- Agile joined with Lean Six-Sigma.
- Agile joined with TSP and PSP.
- Agile joined with the CMM and CMMI.
- CMM and CMMI joined with Six-Sigma.
- CMM and CMMI joined with International Standardization for Organization (ISO) standard certification.
- CMM and CMMI joined with Information Technology Infrastructure Library (ITIL).
- CMM and CMMI joined with Quality Function Deployment (QFD).
- CMM and CMMI joined with TSP and PSP.
- XP joined with Lean Six-Sigma.
- ITIL joined with Six-Sigma.
- ISO joined with TickIT.
- OOD joined with service-oriented architecture (SOA).
- Six-Sigma joined with QFD.
- Six-Sigma joined with ITIL.
- Six-Sigma joined with TSP/PSP.
- SOA joined with ITIL.
- TickIT joined with Six-Sigma.

The list above only links pairs of development methods that are joined together. From time to time three or even four or five development practices have been joined together:

- Agile joined with OOD, joined with Lean Six-Sigma, and joined with ITIL.
- CMM joined with Agile, joined with TSP/PSP, joined with Six-Sigma, and joined with QFD.

	Agile	CMM Level 3	Difference
Size in function points	1,000	1,000	0
Size in Java code statements	50,000	50,000	0
Monthly burdened cost	$7,500	$7,500	0
Work hours per month	132	132	0
Project staff	5	7	2
Project effort (months)	66	115	49
Project effort (hours)	8,712	15,180	6,486
Project schedule (months)	14	19	5
Project cost	$495,000	$862,500	$367,500
Function points per month	15.15	8.67	-6.46
Work hours per function point	8.71	15.18	6.47
LOC per month	758	435	-323
Function point assignment scope	200	143	-57
LOC assignment scope	10,000	7,143	-2,857
Cost per function point	$495	$863	$368
Cost per LOC	$9.90	$17.25	$7.35
Defect potential	4,250	4,500	250
Defect potential per function point	4.25	4.50	0.25
Defect removal efficiency	90%	95%	0.5%
Delivered defects	425	225	-200
High-severity defects	128	68	-60
Delivered defects per function point	0.43	0.23	0.20
Delivered defects per thousand LOC	8.50	4.50	-4.00

Table 2: *Comparison of Agile and CMM Results for an Application of 1,000 Function Points*

3. Acceptance testing of application.
4. Total cost accumulation.
5. Quality data accumulation.
6. Final Scrum session.

These are the most interesting and unique features of the Agile methods: 1) The decomposition of the application into separate iterations, 2) The daily face-to-face contact with one or more user representatives, 3) The daily *Scrum* sessions to discuss the backlog of work left to be accomplished and any problems that might slow down progress. Another interesting feature is to create the test cases before the code itself is written, which is a feature of XP.

Comparative Results of Agile and CMM for an Application of 1,000 Function Points

There is much more quantitative data available on the results of projects using the CMM than for projects using the Agile methods. In part this is because the CMM and CMMI include measurement as a key practice. Also, while Agile projects do estimate and measure, some of the Agile metrics are unique and have no benchmarks available as of 2007. For example some Agile projects use *story points*, some use *running tested features*, some use *ideal time*, and some measure with *use case points*. There are no large collections of data using any of these metrics nor are there reliable conversion rules to standard metrics such as function points.

For this article, an artificial test bed will be used to examine the comparative results of Agile and the CMM. The test bed is based on a real application (an online survey tool), but the size has been arbitrarily set to exactly 1,000 function points or 50,000 Java statements.

A sample of 20 CMM projects examined by the author and his colleagues was condensed into an overall aggregate for the CMM results. For the Agile data, observations and discussions with practitioners on five projects were used to construct the aggregate results. This is not a particularly accurate and reliable way to perform comparative analysis. As it happens, none of the 20 CMM applications used Agile methods, nor did any of the Agile projects use aspects of the CMM or CMMI.

Merging aspects of the Agile and CMM concepts is not difficult. Indeed, some CMM and CMMI applications circa 2007 do use Agile features such as Scrum sessions. However, among the author's clients, about 85 percent of CMM/CMMI users do not use Agile and about 95 percent of Agile users do not make use of either the CMM or CMMI. For example, at a recent conference of more than 100 state software managers, more than 50 percent of current projects were using Agile methods – but no projects at all were using either the CMM or CMMI.

Though briefly discussed in the sidebar, the topic of hybrid approaches is not well covered in the software engineering literature. Neither is the topic of hybrid approaches well covered in terms of benchmarks and quantitative data, although some hybrid projects are present in the International Software Benchmark Standards Group data. To simplify the results in this article, the Agile methods were used in *pure* form as were the CMM and CMMI methods. Since the *pure* forms still outnumber hybrid approaches by almost 10-to-1, that seems like a reasonable way to present the data.

The sizes of the samples were not, of course, exactly 1,000 function points. They ranged from about 900 to almost 2,000 function points. Not all of the samples used Java either. Therefore the following comparison in Table 2 has a significant but unknown margin of error.

What the comparison does provide is side-by-side data points for a standard test bed, with the measurements for both sides expressed in the same units of measure. Hopefully future researchers from both the Agile and CMM/CMMI camps will be motivated to correct the results shown here, using better methods and data.

Note that the *lines of code* (LOC) metric is not reliable for economic studies since it penalizes high-level programming languages and cannot be used for cross-language comparisons. However, since many people still use this metric, it is provided here with the caveat that none of the LOC values would be the same for other programming languages such as C++, Smalltalk, Visual Basic, etc.

The data in Table 2 is expressed in terms of function point metrics and assumes version 4.2 of the counting rules published by the International Function Point Users Group.

The CMM version of the project is assumed to be developed by an organization at Level 3 on the CMM. The Agile version of the project is assumed to be developed by an experienced Agile team. However, the Agile version is assumed not to use either the CMM or CMMI and so has no level assigned. This is a reasonable assumption because very few Agile teams are also CMM certified.

Although one comparison is probably insufficient, observations of many

small projects in the 1,000 function point size range indicate that the CMM and CMMI are somewhat more cumbersome than the Agile methods and therefore tend to have somewhat lower productivity rates (see Table 2). That is not to say that the CMM/CMMI methods produce bad or unacceptable results, but only that the Agile methods appear to generate higher productivity levels.

However, if the size plateau for comparison is upped to 10,000 function points or 100,000 function points, the CMM/CMMI methods would pull ahead. At 10,000 function points the overall staff would be larger than 50, while for 100,000 function points the overall staff would approach 600 people and some of the Agile principles such as daily team meetings would become difficult and perhaps impossible.

Also, applications in the 10,000 to 100,000 function point size range tend to have thousands or even millions of users. Therefore it is no longer possible to have direct daily contact with users since their numbers are too large.

When quality is considered, the Agile approach of having frequent daily contacts with users, daily Scrum sessions, and writing the test cases before the code has the effect of lowering *defect potentials*. The phrase defect potentials refers to the total number of defects that might be encountered in requirements, design, coding, user documents, as well as *bad fixes* or secondary defects. (The current U.S. average would be about 5.0 defects per function point for defect potentials.) However, Agile projects usually do not perform formal design and code inspections so their defect removal efficiency is somewhat below the CMM example.

The defect potentials for the CMM version are better than U.S. averages but not quite equal to the Agile version due to the more detached connections between clients and developers.

However, the CMM and CMMI methods typically do utilize formal design and code inspections. As a rule of thumb, formal inspections are more than 65 percent efficient in finding bugs or defects, which is about twice the efficiency of most forms of testing, many of which only find about 30 percent of the bugs that are actually present.

As of 2007 the current U.S. average for cumulative defect removal efficiency is only about 85 percent, so both the Agile and CMM examples are better than U.S. norms. The CMM method is somewhat higher than the Agile method due to formal inspections, testing specialists, and a more formal quality assurance approach.

As a general rule, methods developed to support large software applications such as the CMM and CMMI are cumbersome when applied to small projects. They can be tailored or subset to fit small projects, but out of the box they are cumbersome.

On the other hand, methods developed to support small software applications such as the Agile methods become less and less effective as application sizes increase. Here too, they can be tailored or modified to fit larger projects, but out of the box they are not effective.

As of 2007 both the Agile and CMM/CMMI communities are working on merging the more useful features of both sets of methods. Agile and the CMM/CMMI are not intrinsically opposed to one another, they merely started at opposite ends of the size spectrum.

Overall Observations on Software Development

With more than 13,000 projects examined, it can be stated categorically that there is no single method of development that is universally deployed or even universally useful. All software projects need some form of gathering requirements, some form of design, some kind of development or coding, and some forms of defect removal such as reviews, inspections, and testing.

In the author's studies more than 40 methods for gathering requirements, more than 50 variations in handling software design, more than 700 programming languages, and more than 30 forms of testing were noted among the projects examined. Scores of hybrid methods have also been noted.

Both the Agile methods and the CMM/CMMI methods have added value to the software industry. But care is needed to be sure that the methods selected are in fact tailored to the size range of the applications being constructed.◆

References

1. Humphrey, Watts S. Managing the Software Process. Reading, MA: Addison-Wesley Longman, 1989.
2. Cockburn, Alistair. Agile Software Development. Boston, MA: Addison Wesley, 2001.

Additional Reading

1. Caputo, Kim. CMM Implementation Guide. Boston, MA: Addison Wesley, 1998.
2. Garmus, David and David Herron. Measuring the Software Process: A Practical Guide to Functional Measurement. Englewood Cliffs, NJ: Prentice Hall, 1995.
3. Jones, Capers. "Defense Software Development in Evolution." CrossTalk Nov. 2002.
4. Jones, Capers. Software Assessments, Benchmarks, and Best Practices. Boston, MA: Addison-Wesley Longman, 2000.
5. Jones, Capers. Estimating Software Costs. New York. McGraw Hill, 2007.
6. Jones, Capers. Conflict and Litigation Between Software Clients and Developers. Burlington, MA: Software Productivity Research (SPR), Inc., 2007.
7. Kan, Stephen H. Metrics and Models in Software Quality Engineering. 2nd ed. Boston, MA: Addison-Wesley Longman 2003.

About the Author

Capers Jones is currently the chairman of Capers Jones and Associates, LLC. He is also the founder and former chairman of SPR, where he holds the title of Chief Scientist Emeritus. He is a well-known author and international public speaker, and has authored the books "Patterns of Software Systems Failure and Success," "Applied Software Measurement," "Software Quality: Analysis and Guidelines for Success," "Software Cost Estimation," and "Software Assessments, Benchmarks, and Best Practices." Jones and his colleagues from SPR have collected historical data from more than 600 corporations and more than 30 government organizations. This historical data is a key resource for judging the effectiveness of software process improvement methods. The total volume of projects studied now exceeds 12,000.

Software Productivity Research, LLC
Phone: (877) 570-5459
Fax: (781) 273-5176
E-mail: capers.jones@spr.com, info@spr.com

Is CMMI Useful and Usable in Small Settings? One Example

Sandra Cepeda
Cepeda Systems and Software Analysis, Inc.

Suzanne Garcia
Software Engineering Institute

Jacquelyn Langhout
Technical Management Directorate

The Software Engineering Directorate (SED) of the U.S. Army Aviation and Missile Research, Development and Engineering Center (AMRDEC) in Huntsville, Alabama, acquires software-intensive systems and has more than 250 small companies in its supply chain. In order to determine the appropriateness of using Capability Maturity Model Integrated (CMMI®) as supplier requirements, members of AMRDEC SED teamed with the Software Engineering Institute (SEI) to perform a technical feasibility study in 2003-2004. This article presents the motivation, the processes used, and the major results of the CMMI for Small Business pilot from the perspective of the team that worked on the pilot.

We often hear that CMMI® *wasn't built for small companies so it will not work for them*, or some variant of this sentiment. Many people find the CMMI book/technical report intimidating to think about using it. Although it is true that CMMI was not explicitly built for small companies, it is also true that it was not explicitly built for large companies [1]. The experience we obtained from the CMMI for Small Business pilot indicates that CMMI, when applied in a way that responds to the business realities of a small business, can provide small companies with utility.

Small Pilot Company Profiles

Two small companies from Huntsville, Alabama were selected to participate in the pilot:

- **Analytical Sciences, Incorporated (ASI)** specializes in management and technical services with a focus on systems engineering/program management, information technology, engineering and scientific services, and professional and organizational development.
- **Cirrus Technologies, Incorporated (CTI)** specializes in manufacturing and support services with a focus on logistics, engineering, manufacturing, test and evaluation, information technology, security, and intelligence.

At the time of the pilot, each company had around 200 employees. The projects selected for the pilot ranged in size from a one-person project to a 22-person project. CMMI v1.1 SE/SW was used as the reference model for the project.

Key Challenges in Process Improvement for Small Business

We saw several challenges during the adoption pilot in Huntsville. Some were challenges that we had hypothesized, some were new insights. Although the pilot was not designed to address all of these challenges, we list them here in the following as a reference to underscore that we acknowledge that there are a diverse set of challenges for CMMI adoption in a small setting:

- Affordability of process improvement is a major challenge.

> *"Ensure that senior management understands how to interpret appraisal results, both in terms of what they are likely to mean in terms of performance and how they can be appropriately used in marketing concepts."*

- Small businesses need to realize payoff quickly.
- Small businesses do not have staff dedicated solely to process improvement implementation: Customer requirements take priority and can cause delays.
- There is minimal structure to leverage from in a small business.
- *The customer rules.* Many small organizations adopt/adapt their business practices directly from their customers or prime contractor.
- If a quality system is either not already in place or is not well-functioning, process definition efforts are much more challenging.
- CMMI is generally perceived as intimidating, both in size and scope.

Motivation for the Pilot

The AMRDEC SED is one of three Life Cycle Software Engineering Centers in the Army. Established in 1984, the SED is a recognized leader in supporting the acquisition, research, development, and sustainment of some of the nation's most sophisticated weapon systems. The mission of the SED is to provide mission critical computer resource expertise to support weapon systems over their life cycle. This mission is carried out by a staff of approximately 900 government and contractor employees housed in the Army's only facility designed specifically for tactical battlefield automated systems support.

Like many federal organizations, the SED relies heavily upon a contract workforce for the fulfillment of its mission. The two primary SED contract vehicles consist of many companies categorized as small businesses. Currently, more than 75 percent of the companies contracted for engineering services with the SED are small businesses. Since these companies are increasingly involved in the development of significant components for software-intensive systems, their usage of reliable engineering and management practices has become increasingly critical to the delivery of quality products for the Department of Defense (DoD) warfighter.

Pilot Process Overview

The CMMI for Small Business pilot started in July 2003 and culminated in May 2004 with a Standard CMMI-based Appraisal Method for Process Improvement v1.1 (SCAMPI℠) Class A appraisal of each of the two pilot companies. The overall process is summarized in Figure 1. Gaps between the organizations' internal processes and CMMI were identified by

℠ SCAMPI is a service mark of Carnegie Mellon University.

engaging in a collaborative session between the pilot team consultants and the practitioners from the pilot companies that was similar to a SCAMPI C appraisal. Based on the results of this analysis, the pilot companies developed and implemented an action plan and updated existing processes to close the gaps found. Where necessary, the pilot companies also developed new processes. Though we initially did not intend to perform SCAMPI A appraisals, the progress made by both companies was such that in January of 2004 we defined appraisal scopes in conjunction with the pilot companies, and in May 2004 we performed SCAMPI v1.1 A appraisals using the continuous representation of CMMI-SE/SW v1.1 at both sites [2]. Both companies achieved their target level profiles, as follows:

- **ASI:** Capability Level 2 for project planning, requirements management, and measurement and analysis, and Capability Level 3 for organization process focus and organizational training.
- **CTI:** Capability Level 1 for project planning, requirements management, and project monitoring and control (given some of the other business challenges that CTI was facing at the time of the pilot, establishing Level 1 processes in these areas was a significant achievement).

Lessons Learned From the Pilot

There are several competencies in process improvement that provide a useful framework for looking at lessons learned from the pilot study. Four of these are included here as a way to organize lessons learned [3]:

- Establishing and sustaining sponsorship.
- Developing infrastructure/defining processes.
- Deploying new processes into the intended use environment.
- Managing an appraisal life cycle.

We have included an additional category of lessons learned in this section: lessons about the CMMI model itself. Those readers who are experienced in process improvement consulting in a variety of settings may recognize our primary competencies as categories that also apply to larger organizations. However, the particular lessons that have been included here are those that we believe are either unique to the small settings environment or are particularly important for a small company to be successful in their improvement efforts.

Establishing and Sustaining Sponsorship

Obtaining and sustaining the executive sponsorship necessary to make applying resources to process improvement activities feasible

- **Lesson 1:** Focus CMMI implementation in areas where the connection between the model's content and the Chief Executive Officer's (CEO) business goals are clearest.

  In a small company, sponsorship often means getting the attention of the owner and/or CEO of the company. In this setting, the focus of the CEO is often on a combination of cash flow management and development of the growth of the company. This implies that any process improvement efforts that are presented must be aligned with the particular financial environment and growth goals of the company.
- **Lesson 2:** Even if you do not have strong quantitative results right away, make sure that the senior management gets periodic progress reports that include the qualitative benefits of the improvement effort.
- **Lesson 3:** Ensure that senior management understands how to interpret appraisal results, both in terms of what they are likely to mean in terms of performance and how they can be appropriately used in marketing contexts.

Developing Infrastructure/Defining Processes

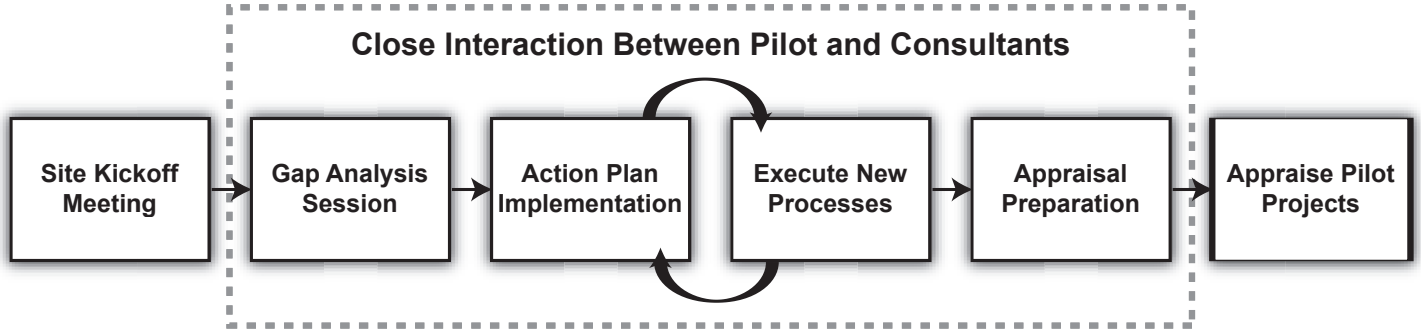
Providing enabling infrastructure to make definition and use of new processes effective

Examples of activities that fit in this category include the following:

- Establishing/managing a process asset library.
- Establishing/managing a measurement repository.
- Establishing/maintaining standards, approaches, and accepted methods for writing process guidance.
- Establishing/managing the organization's curriculum for process improvement.
- Establishing points of contact or specific groups (e.g., an engineering process group [EPG]) for various aspects of the improvement.

- **Lesson 4:** Even though a formal EPG may be infeasible for small companies, some focal point for coordination is particularly needed to coordinate infrastructure development and sustainment.
- **Lesson 5:** When a well-functioning quality management system is already in place (e.g., based on International Organization for Standardization [ISO] 9001), take advantage of it! The existence of a well-functioning ISO 9001-based quality management system provided a bootstrap for process guidance standards and several other elements of process improvement infrastructure. On the other hand, if there had been no quality system already in place, some time would have been needed to establish and set up procedures for using some kind of mechanism for storing, controlling, and distributing process assets created as part of the improvement effort.
- **Lesson 6:** The tools and practices of the accounting system have a great influence on what is considered doable

Figure 1: *Pilot Process Overview*



in terms of collecting and using measurements. A small company typically does not have the resources available to create a parallel metrics collection system from their mainstream accounting system, so, at least at the beginning, what is considered feasible in terms of measurement is constrained by what can be collected/aggregated by the tools in use.

Deploying New Processes Into the Intended Use Environment

Ensuring that the new CMMI-informed processes are available to all relevant users and that their successful adoption is associated with appropriate training and job aids. This is where much of what we traditionally call organization change management occurs

- **Lesson 7:** Simple CMMI-based improvements can have a significant impact in small organizations.

  In one case, just adding meeting minutes for the weekly meeting and publicizing them to the customer and project participants (not more than five people total) contributed to more efficient monitoring of the project and improved communication between the customer and the project team. It sounds simple, and it is: The model provided an incentive to try something so there would be records of decisions/status progress. However, the effect was much greater than the project participants anticipated, both in terms of scope of effect and magnitude – the change not only provided an effective tool for monitoring but it also resulted in improved communication with the customer, which greatly improved the performance of the project as a whole.

  Seeing unanticipated benefits from small changes was a great motivator for continuing on the path of improvement and being willing, a little later in the process, to try larger changes. In small companies, the effects of small changes can often be seen much more quickly and the dispersal of knowledge throughout the company about the effects of a change is also faster.

Managing an Appraisal Life Cycle

Selecting a method of measuring progress against a model (i.e., appraisal method) and then planning and executing the tasks associated with the selected method

- **Lesson 8:** Use a focused, *collaborative* appraisal method (e.g., SCAMPI B or C) for the initial gap analysis. Great benefit is realized by using this session as an opportunity to interpret the model and gain a better understanding of how it applies to the organization.
- **Lesson 9:** Ensure someone in the organization has a good understanding of Appraisal Requirements for CMMI Class A, B, and C appraisal methods and set expectations [4]. This greatly increases the potential for achieving the appraisal objectives defined by the appraisal sponsor.
- **Lesson 10:** Collect evidence that will be useful in the appraisal as you go using automation support as much as possible. Interact with the lead appraiser during evidence collection and mapping to CMMI practices to ensure that a complete, well-organized set of evidence is available for the appraisal. This does not need to be days and days of billable interaction. It may just take the form of e-mailing templates for evidence collection to get an idea of how they fit with the lead appraiser's expectations.

  Although this is one of the lessons that is also equally applicable in a larger setting, the effects if this is NOT done are much greater in a small setting in terms of the percentage of staff time that has to be used to rework material that has been prepared for the appraisal.
- **Lesson 11:** Introduce generic practices once specific practices are clearly understood but prior to the definition and documentation of processes. *Misinterpretation of generic practices is a major cause for appraisal failures.* This is an area where investing in a small amount of external consulting could pay big benefits. In the case of the pilot projects, we held a generic practices workshop to help the pilot participants get a better understanding of the linkages between generic practices and the process areas they were working with.
- **Lesson 12:** Quick looks (e.g., SCAMPI B and SCAMPI C) significantly improve the chances for achieving the objectives of a SCAMPI A.

CMMI Model

- **Lesson 13:** Overall, we saw that judicious use of the elements of CMMI that relate to the business context provided a set of useful practices from which small businesses can benefit, though not always in predicted ways.
- **Lesson 14:** Using the continuous representation of the model allowed the pilots to focus on improvements that they perceived as having the highest payoff for the company.
- **Lesson 15:** Changing the practices in the model is not necessary in most cases; finding alternative practices is often relevant. In addition, work products generated as a result of practice implementation rarely match one-to-one to what is suggested in the model.
- **Lesson 16:** *Smallness* was not as much of an issue for model interpretation as the focus of the business. Although both organizations had a more traditional product development project included in their pilot, they also had more pure service delivery contexts (give me a team of N people who can do X for 25 hours per month for the next six months) that they wanted to explore because service delivery is the heart of their business. Sometimes those services are delivered in the context of a project, but they often are not. The model was more difficult to interpret in areas of the pilot involved in service delivery than in the small product development projects. The SEI is involved in an effort led by Northrop Grumman to develop a CMMI for Services (SVC) constellation that may prove more useful in this context. Information on CMMI-SVC can be found on the CMMI Web site at <www.sei.cmu.edu/cmmi>.

A Toolkit to Help You Start Your Own CMMI-Based Improvement Effort

As a major product of the pilots, the team produced a Web-based toolkit that provides details on the processes and assets used in the pilot. (The draft of the toolkit can be found at <www.sei.cmu.edu/cmmi/publications/toolkit/>. It is a draft that was not fully completed due to budget constraints. It may get incorporated into the Implementing Process Improvement in Small Settings [IPSS] Field Guide, in which case it would be updated.) In addition to process descriptions, it provides copies of the actual presentations, templates, and other documents used to support the pilot. It should be treated as an anecdotal set of assets that might be useful in supporting a model-based improvement effort, rather than a canonical set that defines what *should* be used. Having said that, we believe that the toolkit can help people working on improvement in the following small settings:

- Focus their improvement efforts.
- Figure out how and where to get started.
- Tie their improvements to business goals.
- Educate their staff in areas where they may need to improve their knowledge and skills.
- Realize payoffs (mostly qualitative) early in the improvement effort.
- Improve their ability to prepare for appraisals.

The feedback to date that we have received on the toolkit has been very positive and fairly broad in terms of global access (people from Argentina, Israel, United Kingdom, Mexico, and Chile, as well as the U.S. and Canada have accessed the toolkit).

In thinking about using the toolkit, we have a few recommendations for those who are working in small settings currently and are planning to use it to support your improvement effort:

- Think of this as *one* resource to help you, but not the only one. Every month there are new publications related to CMMI; some of them are likely to offer different insights than the toolkit but they may be valuable to you.
- Be sure to read the *What's Missing* section of the toolkit to see if any of the things we talk about in that section apply to you. If they do, then you know you will need resources beyond what we used to get you started and be successful.
- For those of you who are in the DoD supply chain, think about getting mentoring from the larger companies that work with you and have ongoing improvement efforts; they should have a vested interest in your success.
- Keep up with the assets in the CMMI adoption area (<www.sei.cmu.edu/cmmi/adoption>); that is where you will see emerging work on CMMI in small settings in particular and other resources that may be of value to you.
- Explore at a reasonable pace. Unless you have some business investment riding on achievement of some particular status related to CMMI, do not try to do too much at once until you have established what benefits you can accomplish in your own environment.

Next Steps

SED's Plans for Follow-On Activities

The CMMI small business pilot has been one of the most beneficial endeavors of the SED/SEI strategic partnership. We are pleased that the AMRDEC SED-sponsored pilot provided the stimulus for the establishment of the IPSS project at the SEI. One of the early events of this project was an International Research Workshop in this topic area that was held at the SEI in October 2005 and resulted in an SEI Technical Report summarizing the workshop and containing the papers submitted to the workshop. This report is available for download in the publications section of the SEI Web site [5].

As the SED/SEI partnership continues, we will start to gain insight into the use of some other SEI technologies within the SED setting. These include the insertion of Personal Software Process℠/Team Software Process℠ technology in an Army pilot program to provide the acquisition organization with greater insight into development metrics. Additionally, the SED/SEI partnership serves an integral role in providing acquisition process improvement support to many of our local Army program managers.

> *"For those of you who are in the DoD supply chain, think about getting mentoring from the larger companies that work with you and have ongoing improvement efforts; they should have a vested interest in your success."*

SEI's Plans for Supporting CMMI for Small Settings

The pilot project in Huntsville, Alabama emphasized to the SEI the need for appropriate guidance materials for using CMMI in small settings. In response, the SEI has chartered the IPSS project within the International Process Research Consortium initiative. Seed funding resulted in the International Research Workshop mentioned earlier, and initial sponsors are supporting the prototyping of an IPSS Field Guide that reflects many of the lessons cited here. Contact Caroline Graettinger, the IPSS project manager, for details, at <cpg@sei.cmu.edu>.

Conclusion

We hope you will find this information beneficial as you embark on your own improvement journey and you will become a member of the burgeoning community of practice for CMMI in small settings. Stay tuned with ongoing SEI work in small settings at <www.sei.cmu.edu/iprc/ipss.html>. This endeavor is discussed more on page 27.◆

References

1. Chrissis, Mary Beth, Mike Konrad, and Sandy Shrum. CMMI: Guidelines for Process Integration and Product Improvement. Boston, MA: Addison-Wesley, 2003 <www.sei.cmu.edu/cmmi/publications/cmmi-book.html>.
2. Members of the Assessment Method Integrated Team. Standard CMMI Appraisal Method for Process Improvement, Ver. 1.1: Method Definition Document. CMU/SEI-2001-HB-001. Pittsburgh, PA: SEI CMU, 2001 <www.sei.cmu.edu/publications/documents/01.reports/01hb001.html>.
3. Garcia, Suzanne, and Richard Turner. CMMI Survival Guide: Just Enough Process Improvement. Boston, MA: Addison-Wesley, 2006.
4. CMMI Product Team. Appraisal Requirements for CMMI, Version 1.1 (ARC, V1.1). CMU/SEI-2001-TR-034. Pittsburgh, PA: SEI CMU, 2001 <www.sei.cmu.edu/publications/ documents/01.reports/01tr034.html>.
5. Garcia, Suzanne, Caroline Graettinger, and Keith Kost. Proc. of the First International Research Workshop for Process Improvement in Small Settings. CMU/SEI-2006-SR-01. Pittsburgh, PA: SEI CMU, 2006.

Acknowledgements

Many people contributed significant resources to this pilot. The CMMI in Small Settings Toolkit Repository from AMRDEC SED Pilot Sites Web site, located at <www.sei.cmu.edu/ttp/publications/toolkit>, contains an acknowledgments table that we hope covers most of the people to whom we owe gratitude. We would, however, particularly like to acknowledge the ASI Team and the CTI Team. Without their dedication, this pilot would not have been possible, let alone successful. We are also grateful to Gene Miluk, of the SEI, and Mary Jo Staley, of CSC, who were consultants for the pilot and now have moved on to other endeavors. Their ideas and hard work during the pilot made possible much of the learning reflected here.

About the Authors

Sandra Cepeda, President and CEO of Cepeda Systems and Software Analysis, Inc., is a CMMI consultant, an SEI-authorized lead appraiser for SCAMPI, an SEI-authorized CMMI Instructor, and an SEI Visiting Scientist. Her company provides engineering services to the Army's Aviation and Missile Research Development and Engineering Center, SED. Cepeda's 21 years of experience in complex systems development has spanned all aspects of the life cycle, with a particular focus on Verification and Validation (V&V) and Process Improvement. She was a member of the CMMI 1.1 and CMMI 1.2 and SCAMPI 1.2 author teams, and was one of the lead consultants for the CMMI for Small Business pilot in Huntsville. Cepeda's has a bachelor's and master's degrees in computer engineering from Auburn University.

AMRDEC SED
AMSRD-AMR-BA-SQ
Hackberry RD
BLDG 6263
Redstone Arsenal, AL 35898
Phone: (256) 876-0317
E-mail: sandra.cepeda@us.army.mil

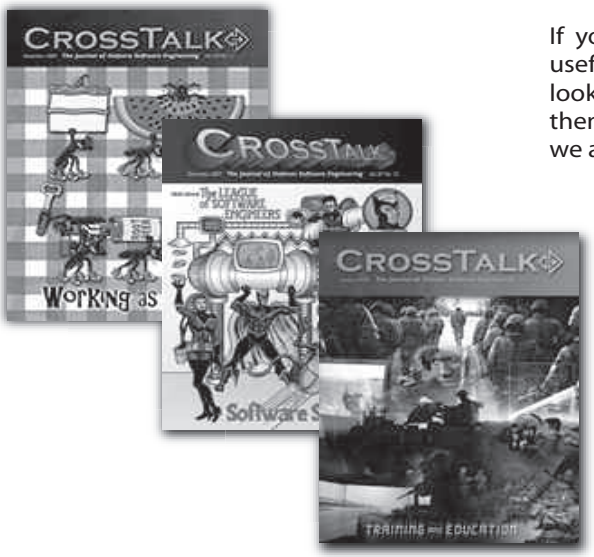
Suzanne (SuZ) Garcia is a senior member of the technical staff at the SEI CMU, working in the Integrating SW-Intensive Systems group. Her current research is focused on synthesizing effective practices from research and industry into effective techniques for use by the software and systems engineering communities working in a system of systems context. Garcia worked in the Technology Transition Practices group, with a particular focus on the technology adoption issues related to small settings. Her early SEI career focused on authoring, managing, and reviewing CMMs, followed by three years as the deployments manager for Aimware, Incorporated's U.S. customers. Garcia also spent 12 years in multiple improvement-related roles at Lockheed Missile and Space Co. She has a bachelor's degree and a master's degree in systems management.

SEI CMU
Pittsburgh, PA 15213-3890
Phone: (412) 268-9143
E-mail: smg@sei.cmu.edu

Jacquelyn (Jackie) Langhout is currently the deputy director of the Technical Management Directorate at the AMRDEC at Redstone Arsenal, AL. During the time of the pilot, Langhout served as the lead of the Engineering Process Group at AMRDEC's SED. In that role, Langhout was the focal point for all the SED-SEI strategic partnership efforts as well as overseeing the SED's process improvement program. Her past assignments include leading software V&V projects, software development and maintenance projects, and providing software support to program offices. Langhout began her career with the Army in 1986 and is a member of the Army Acquisition Corp. She has a bachelor of science in mathematics from Samford University, a bachelor's degree from Auburn University, and a master's degree from the University of Alabama in Huntsville.

AMRDEC
AMSRD-AMR-TM
BLDG 5400 RM S142
Redstone Arsenal, AL 35898
Phone: (256) 876-4182
E-mail: jackie.langhout@us.army.mil

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:

20th Anniversary Issue
August 2008
Submission Deadline: March 14, 2008

Application Security
September 2008
Submission Deadline: April 18, 2008

Development of Safety Critical Systems
October 2008
Submission Deadline: May 16, 2008

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>.

Why Do I Need All That Process? I'm Only a Small Project©

Mark Brodnik, Robyn Plouse, and Terry Leip
Intel Corporation

At Intel's Information Technology (IT) department, we developed extensive processes for our projects. While the large projects get the glory, the majority of our projects are less than six months long, have small teams, limited scope, and low risk. We found that we have a variety of project sizes but a single set of processes originally built for larger projects. So how did we fix that issue?

The Intel IT department, like many other organizations, was seeing issues with completing software projects on time and satisfying our customers. Using the Software Engineering Institute's Capability Maturity Model Integration (CMMI℠) as a basis to tackle those issues, management challenged us with the goal of being at Maturity Level 3 in 18 months. While we acknowledged that this goal was overly ambitious, we still took a shot at accomplishing the objective. A small team using the CMMI as their guide built large and formal processes and mandated their use. The inevitable result was that this approach was not well received or implemented by the project teams, especially those at the smaller end of the scale. The IT organization also realized that there was no business justification for a CMMI benchmark. To address this issue, we launched a project to streamline the processes; we took a different look at the CMMI, listened to the stakeholders, and focused on our business objectives. The end result was a set of processes 70 percent shorter and much less prescriptive.

Accelerated Process Improvement (API)

Based on stakeholder feedback, audit results, and process coach input, we identified a number of the work processes that were complex and difficult to use. The organization gathered 35 representatives including key stakeholders, engineers, process coaches, and auditors for an intense three-and-a-half day face-to-face meeting. In preparation for the session we mapped business and model requirements to our existing processes in order to determine what portions were eligible for simplification. In addition, we created a priority list of feedback from audit, coaching, and pending process improvement requests to determine top *pain points* and focus areas. Since we had limited face-to-face time to complete the work, we were willing to accept less than perfect results and less adherence to the model to streamline the business process.

A key component of the successful collaboration was sequestering the team at an off-site facility to maximize focus and support a collaborative environment. The agenda for this event was driven by the stakeholders rather than the process engineers. The stakeholders drove the vast majority of the changes with a primary focus on accomplishing business results rather than textbook model compliance. The team reviewed each work process against the business requirements and the CMMI model to identify portions of the process that could be considered for deletion or simplification. The group was broken down in to three sub-teams each addressing a different process area. Daily report out sessions and synchronization sessions helped ensure consistency between the teams. Our CMMI model expert floated between the teams to answer model related questions.

One of the themes we noticed was that the initial process development team had implemented many of the sub-practices in the CMMI model believing they were a required model component. For example, based on sub-practices from the configuration management process area, we required each document to have unique identifier labels both internal to the document and in the file name – a convention that was unnecessarily complex and added little value for our typical 8-to-12 person project teams. We also defined and mandated specific elicitation techniques to meet the intent of the sub-practices in the requirements development process area; these techniques were overkill for our small project teams which worked closely with their customers and stakeholders. Finally, we eliminated the requirement for a separate formal commitment for resources by combining this with one of the key milestone decisions. This came about by an incorrect interpretation of the sub-practices in the project monitoring and control process area.

Figure 1 (see page 20) shows the old processes for scope, schedule, and resource management. These separate processes totaled 29 pages and 17 process steps. The new process was able to combine scope, schedule, and resource management into one process document that was five pages long with eight process steps. This was accomplished by simplifying the wording, combining or eliminating process steps, and removing instructional text which we thought would substitute for skills expertise from the process documents.

Overall, we achieved a 70 percent reduction in document length, with one process shrinking to just 17 percent of its original size. Detail of other process and template reductions can be seen in Table 1 (see page 20). The original and combined processes can be accessed via the online version of this article.

When we evaluated the outcome of the effort, one of our project managers said the following:

> The API face-to-face session was great for analyzing the weak spots (in the documentation and in the processes), as well as acknowledging the flaws of the previous release. The teams addressed the issues with great teamwork that allowed for development of some creative and tangible action plans.

In the end, we reduced the workload of our processes and improved documentation so that it would more effectively deliver key information. The result of process simplification was the ability of smaller projects, originally deemed out of scope,

™ Intel, the Intel logo, Intel. leap ahead., and Intel. Leap ahead. logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
* Other names and brands may be claimed as the property of others.
© 2007 Intel Corporation. All rights reserved.

This article is for informational purposes only. INTEL MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS ARTICLE.

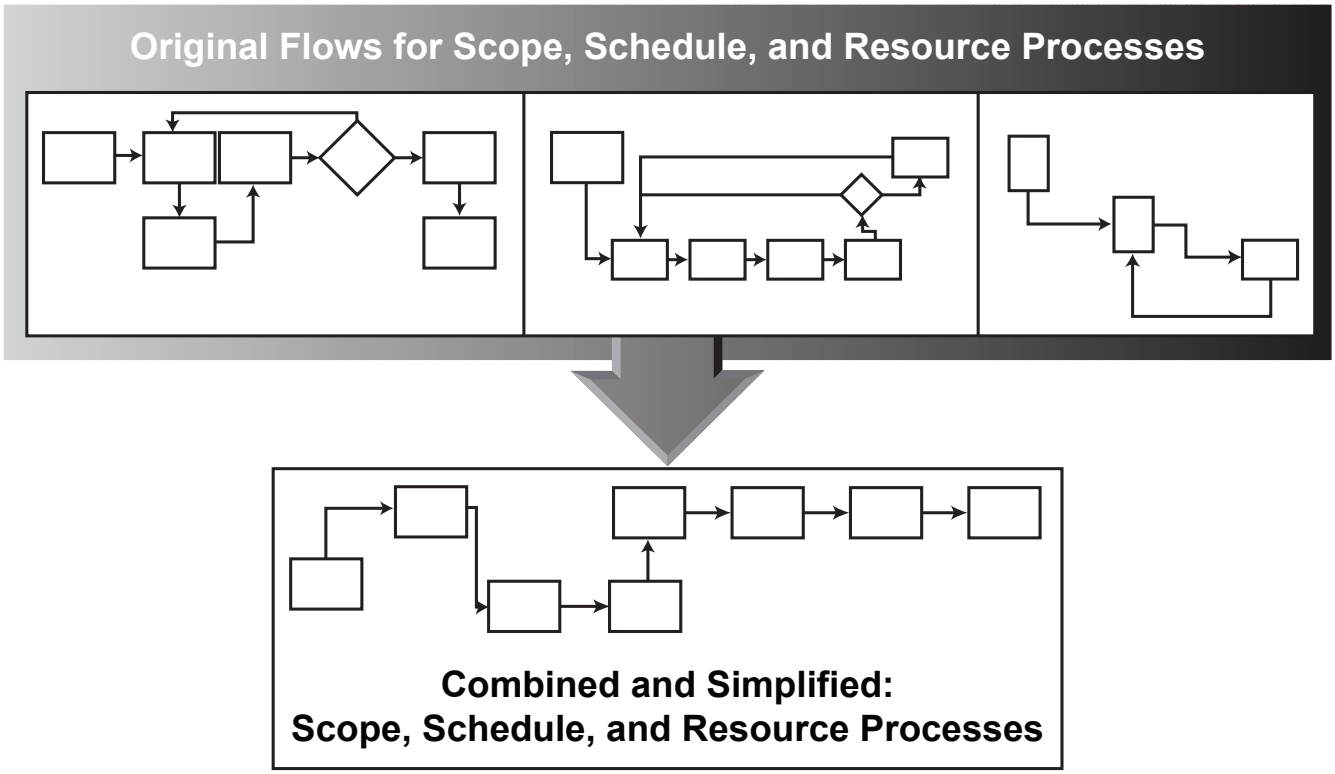


Figure 1: *Example Simplification and Reduction*

to adopt the processes. Our number of eligible projects went from 36 percent to 50 percent. Our process coaches could now support an average of 18 projects per coach, up from 14 projects per coach before the simplification. In addition, the training course delivery was streamlined to half of their original duration.

What About Very Small Projects?

As senior management started seeing the benefits of standard processes, they made the decision to require all projects greater than eight weeks in duration to adopt the processes. As the adoption rates for these projects came closer to 100 percent, we began to look for ways to help the projects originally deemed too small for our processes. These very small projects, less than eight weeks in duration, were considered low risk and therefore were not required to follow any formal processes or standards, although some did have their own self-imposed methods of running their projects. A result of this was that management did not have sufficient visibility into these projects and some of the larger projects would attempt to *chunk* their work into less than eight weeks so they could avoid the full process and external visibility into their work.

In our discussions with project managers, we also realized that the eight-week rule was too limiting, since many projects were only slightly longer (8 to 12 weeks), but had just two or three fractional resources. These project managers felt that the overhead of the processes were still too large compared to the amount of overall effort of the project, and they wanted to also take advantage of a smaller set of processes. They argued that they spent most of their time filling out forms or compliance checklists rather than focusing on the task at hand. We also discovered through audits that project managers in this class of projects would often run their projects as usual and then go back after the fact and create the required process collateral. Clearly, there was need to provide processes that addressed very small projects.

Table 1: *Old Versus New Document Sizes*

Document	Old Size	New Size
High-Level Requirements Process	9 pages – 7 steps	4 pages – 5 steps
Detailed Requirements Process	11 pages – 6 steps	4 pages – 5 steps
High-Level Requirements Template	10 pages	2 pages
Detailed Requirements Template	14 pages	6 pages
Scope/Schedule/Resource Process	29 pages – 17 steps	5 pages – 8 steps
Project Plan Plus Supporting Documents	4 documents	1 spreadsheet
Baseline Change Control Process	6 pages – 13 steps	5 pages – 9 steps

Proof of Concept (PoC)

Before we deployed a new set of processes across all of IT, we decided the best approach was to work with a limited set of these very small projects through a POC in one or two IT groups. We would then analyze the data to determine if an IT-wide deployment was warranted or if we needed additional pilots. Finally we planned to take a thorough look at our entire process library to implement a more robust project characterization approach that right-sizes the processes and tools used by our project teams.

The first area we looked at was how our process deliverables were organized. Our projects' process deliverables are defined at two main levels. The first level is what a project needs to review at a milestone decision meeting with management (ex. requirements, design, and schedule). The second level is deliverables that are typically produced as a result of project work (e.g., various supporting plans, internal compliance scorecards, review and approval of records, change records). In order to address the issue of excessive process overhead for small projects, the

approach to the PoC was to determine the major process deliverables that a project should produce and require only those deliverables. To speed up the PoC, we started with the first-level deliverables and decided to leave the creation of second level process deliverables up to the discretion of the project manager. Ultimately, the goal was to have project managers focus more on the product deliverables and less on the process deliverables.

The next item we looked at was combining the templates for the various process deliverables into a single, simplified template that contains the basic information the projects are required to complete. This allowed decision makers to review this document at milestone decisions rather than requiring the project manager to duplicate information in a separate, formal presentation.

The last major challenge we wanted to work through in the PoC was the cutoff point between these very small projects and our larger projects. In the end, we decided that a more pragmatic approach was to look at several attributes to consider for the cutoff such as team size (one to three team members), overall effort (around 500 hours), and project risk (no financial systems or software exchange impact) to provide a more balanced set of classifications.

Results

While the model is not our primary focus, we are still measuring ourselves against its requirements to make sure we have not missed a best practice. Through a series of Standard CMMI Assessment Method for Process Improvement℠ (SCAMPI) Cs, Bs, and As performed over the past three years we are seeing increased compliance with the model. We have been measuring progress toward Capability Level 3 in Configuration Management, Project Planning, Project Management and Control, Requirements Management, and Requirements Development, and we are also beginning to measure compliance for Measurement and Analysis, Process and Product Quality Assurance, and Verification and Validation.

From a business perspective, we are seeing positive results in shortening the overall duration of our projects and improving our ability to meet our committed delivery dates. This is done by paying particular attention to the planned versus actual time in each individual phase.

In addition, our quality assurance audit team discovered that process compliance had improved as a result of the simplification. Since the processes had changed so dramatically, it was not possible to do an apples-to-apples comparison of the audit results prior to simplification, but both auditors and project team feedback indicated that compliance was better and easier to achieve.

Summary

In our process engineering journey we learned the following things:

- First and foremost, we need to get our stakeholders involved in every step of the development process. This ensures quicker acceptance by project teams and uncovers usability problems earlier in the process.
- Second, it is essential to have a business focus rather than a model focus. It is easy to lose sight of business objectives when trying to comply with CMMI requirements.
- Third, we should start by building processes as small as possible and add to them only as needed. We learned that by building large processes first, the true cost extends beyond the development and requires significant effort to support and maintain.
- Finally, time is the most precious commodity in process improvement. Getting a quick win that the organization would accept is more important than perfection.

In the end, it is all about being both efficient and effective.◆

About the Authors

Mark Brodnik, Project Management Professional (PMP), is a project manager at Intel's IT group where he specializes in process development and process improvement. He has 15 years of IT software applications development and management experience and holds a bachelor of science degree in management information systems from the University of Arizona.

Intel Corporation
M/S C11-123
6505 W Chandler BLVD
Chandler, AZ 85226
Phone: (480) 552-7028
E-mail: mark.brodnik@intel.com

Robyn Plouse, PMP, is a SCAMPI lead appraiser and a quality auditor at Intel's IT group where she also leads the CMMI Community of Practice. She has 10 years of project management and quality experience with Intel and nine years of experience as a systems engineer at Lockheed Martin. She holds a bachelor of science degree in aerospace engineering from San Diego State University and a masters degree in engineering management from Santa Clara University.

Intel Corporation
M/S RR5-505
1600 Rio Rancho BLVD S.E.
Rio Rancho, NM 87124
Phone: (505) 893-6319
E-mail: robyn.plouse@intel.com

Terry Leip is the lead quality assurance auditor at Intel's IT group. He has more than 20 years of software development and quality experience with Intel and Lockheed Martin and holds a bachelor of science degree in biology and computer science from Grand Canyon University.

Intel Corporation
M/S C11-123
6505 W Chandler BLVD
Chandler, AZ 85226
Phone: (480) 552-2079
E-mail: terry.leip@intel.com

Small Project Survival Among the CMMI Level 5 Big Processes

Alan C. Jost
Raytheon

In the past several years, many engineering firms have stepped up and achieved Capability Maturity Model Integrated (CMMI) Level 5 [1]. *Currently, 13.9 percent of the 2,140 appraised organizations have achieved CMMI Level 5 using the Standard CMMI Appraisal Method for Process Improvement (SCAMPI). To achieve this maturity level, the organization must implement a rigorous process definition to address the CMMI model through maturity Level 5. Once defined, the organizational process must be implemented throughout the organization even by small projects. I will examine how, in my business unit, we have successfully achieved CMMI Level 5 across five geographically dispersed business units and how small projects have survived within our significantly large process definition.*

The Raytheon Network Centric Systems (RNCS) has five major engineering centers across the country. An interesting footnote is these five major engineering sites had five completely different process definitions because before they were part of a common business unit, they were actually competing companies. The Northeast location in Marlborough, Massachusetts where I work, was part of the original Raytheon. The St. Petersburg, Florida, location was originally part of E-Systems. The Fullerton, California, location was part of Hughes Defense. The Fort Wayne, Indiana, location was also part of Hughes, but a different internal organization. The McKinney, Texas, location was part of Texas Instruments. Each had their own organizational process definition and in the past each had conducted separate Capability Maturity Model evaluations and CMMI appraisals ranging from maturity levels 3 to 5 and for a variety of discipline combinations: software or software-systems.

Raytheon has defined and implemented a company-wide Integrated Product Development System (IPDS) that provides a program-level process for all Raytheon programs. IPDS provides an extensive process definition describing what should be done throughout a program; the organizations managing programs had to define processes on how IPDS was going to be implemented in their programs. These organizational processes were the ones the RNCS organizations used in their Standard Capability Evaluation (SCE) or SCAMPIs and, of course, they were all different. Then the enlightened senior management of RNCS made a crucial business decision that all five sites were going to have a common engineering implementation of IPDS with the ultimate goal of having the capability to move work across the five sites. The five sites endeavored over a couple of years to define, deploy, and implement the RNCS Common Process Architecture (CPA). The effort ultimately led to the five major engineering centers successfully conducting our SCAMPI that resulted in a Maturity Level 5 rating reported to the Software Engineering Institute (SEI) as *Raytheon Network Centric Systems (Engineering – software engineering, software, hardware [SE/SW/HW]) projects with engineering development in the scope of the project* [2].

We included hardware amplifications in order to include in the appraisal our hardware engineering discipline. In addition to – and key to – our success was our internal program engineering discipline. The key to our successful SCAMPI was the definition and deployment of the organizational CPA across the five RNCS sites. Let us examine this CPA organizational process definition.

Figure 1: *Introduction to WI*[1]

Work Instruction — Number U0141CRP; Revision B; Supersedes A; Date 01/22/07

Detailed Planning

Introduction

The purpose of the Detailed Planning WI is to direct a program's detailed engineering planning activities based on the initial planning outputs and the scope of the awarded contract as part of preparing for the program's Gate 5. During the detailed planning activities those work products from the initial planning activities are reviewed, and updated or expanded as necessary. Any planning work products that may have not been produced as a result of the initial planning activities, but are determined to be required after contract negotiations will need to be created as part of the detailed planning activities.

The complete set of detailed planning activities is defined by this WI and the supporting Plan Generation Work Instructions for the various program plans. Refer to *Appendix A* for an overview of the plans hierarchy. A program can use the NCS Engineering process to tailor the engineering plans identified in *Appendix A* based on the needs of the program. The program plans should be put under configuration control per the NCS Work Product Management Work Instruction.

Input(s)

- Contract Documents
- Engineering Cost Estimates
- IPDS Tailoring
- NCS Engineering Tailoring
- Program Organization Structure
- Initial Engineering Program Plans
- Lessons Learned Repository
- Process Assets Library
- Work Breakdown Structure (WBS)
- Engineering Budget

CPA

The design of the CPA followed the typical life cycle that our programs follow. We initially started creating CPA with a requirements definition phase where we used our Raytheon IPDS, the CMMI model, and International Organization for Standardization (ISO) 9000, to provide the requirements for the CPA. As we defined our CPA, the identified requirements were entered in our requirements database and then were flowed down to organizational behavior activities. I am not going into the details of the overall CPA process, which is not the focus of this article. However, to make a very long story short, the main artifacts of the CPA definition were a series of CPA Work

Instructions (WI). There are 88 WIs to be exact. Each of these has associated with it enablers to assist in process implementation, such as the following: software tools, templates for required documents, process checklists, and other enablers to be used by all programs at the five engineering sites. Naturally, the CPA organizational process definition is substantial in detail and volume, but one of the major cost advantages is that while it consists of a large amount of documentation, it pales in size to having five different sites each with their own process definition, set of tools, templates, and enablers. It was common and had to be used by all programs. The CPA has to be robust enough to support programs with engineering staffs of more than 100 engineers such as the next generation destroyer for the Navy or the replacement of all the Air Traffic Control systems in the United States. Well, CPA is robust enough not only to support these substantial programs being implemented across the five engineering sites, but also supports CMMI Maturity Level 5 processes.

What about the programs that do not have a marching army of engineers with budgets in the millions and billions that are executed over multiple years? How do these small programs deal with the substantial organizational process that was formally appraised at CMMI Level 5 and now required by all RNCS programs? It can be wrapped up in one word: tailoring. Tailoring is a key organizational process that is used by large, medium, and small programs. All programs can eliminate WIs and can accept, reject, or modify the WI requirements as well as modify the contents of the plans as needed by each program's contracts; it is the extent and depth of the tailoring that allows the small programs to survive. However, before getting into tailoring, it is important to understand the general structure of the CPA WIs.

CPA WI Structure and Tailoring

Every WI has the same format structure (see Figures 1 through 4, each one will be detailed in this section).

Figure 1 is the first part of the CPA WI that provides an introduction to the WI process, WI name and configuration management information of the WI. It also provides the input necessary for the process. This particular WI is the main WI used by the typical large program to perform the detailed planning of the program during the program start-up phase that occurs right after contract award. The Detailed Planning WI defines all the steps that the program management team accomplishes in the approximately 45 days after the contract is signed. In actuality, some of the Detailed Planning WI steps require updates to documents and other artifacts previously drafted during the Initial Planning WI steps that were accomplished during the proposal stage of the project. From the Initial Planning WI there are several outputs that are also inputs to the Detailed Planning WI, for example: Initial Engineering Program Plans and Work Breakdown Structure (WBS). Typically, several of the Initial Engineering Program Plans (i.e., Software Development Plan [SDP], System Engineering Plan [SEP]) are drafted during the proposal stage and in the beginning of the program after contract award during program start-up, they are reviewed and finalized. Likewise, the WBS is generated during the proposal time and may have to be adjusted depending on the results of contract negotiations. A portion of the Detailed Planning WI is shown in Figure 2.

Process Summary	ID	Steps at a Glance	Responsible Role
_	1	Define Planning Strategy for Detailed Planning	Program Engineer Program Planning Team
_	2	Update the SEP, HDP, SDP, and QPP	Program Engineer Program Planning Team
_	3	Update Program's Life Cycle Model	Program Planning Team

Note: SEP – System Engineering Plan, HDP – Hardware Development Plan, SDP – Software Development Plan, and QPP – Quality Program Plan.

Figure 2: *Detailed Planning Steps at a Glance*[2]

In Figure 2, you see the first three *requirements* or steps at a glance in the 33 steps in the detailed planning process. Interestingly, these Steps at a Glance were flowed-down as requirements from IPDS, CMMI, and ISO9001 into the detailed planning behavior process (Figure 3).

Figure 3 shows the expected outputs from when the CPA Detailed Planning WI implemented the detailed planning process, which included multiple engineering plans and the WBS. These are artifacts produced or updated as the detailed process is implemented. Most of the plans generated from this work instruction are updates to those drafted in the initial planning and proposal phase. Each of the plans is enabled through the use of templates with

Figure 3: *Outputs From the Detailed Planning Process*[3]

Output(s)
Earned Value Management System (EVMS) baseline
Engineering Cost Estimates
Engineering Decision Analysis and Resolution Plan
Engineering Measurement, Analysis and Improvement Plan
Engineering Training Plan
Work Project Management Plan (WPMP)
Gate Review Plan
Hardware Development Plan
Integrated Master Plan
Integrated Master Schedule
Integration, Verification and Validation Plan
IPDS Tailoring
NCS CPA Tailoring
Planning Strategy
Program Organization Structure
Program's Security Plan
Quality Program Plan
Risk and Opportunity Management Plan
Software Development Plan
Systems Engineering Plan
WBS

Update Program's Life Cycle Model	**Requirement** Update and document the development life-cycle model that will be used. **Guidance** 1. Review and update as necessary the program's life cycle phases on which to scope the detailed planning effort. Refer to the Life Cycle Models in the Raytheon Process Assets Library (RayPAL), for descriptions of candidate models. See *References* section for further information on accessing these models in RayPAL. 2. Document the life cycle phases in the applicable program plans such as the SEP, SDP, or Hardware Development Plan (HDP). 3. Update the systems engineering capabilities expected at the end of each applicable life cycle phase in the SEP. 4. Update the software engineering capabilities expected at the end of each applicable life cycle phase in the SDP. 5. Update the hardware engineering capabilities expected at the end of each applicable life cycle phase in the HDP.

Figure 4: *Step 3 of Detailed Planning Work Instruction*[4]

embedded instructions on what information should be included in the plans (see Figure 4).

Figure 4 shows one of *33 Steps at a Glance* in the Detailed Planning WI. It shows how each step is expanded into the CPA process requirement under the *Requirement* heading along with its associated Guidance on how to implement the Step at a Glance CPA requirement. Naturally, there is a WI for tailoring the WIs, which in itself can also be tailored (see Figure 5). Along with the Tailoring the CPA Process WI, the organization has provided a Guidance enabler on tailoring that includes guidance for tailoring small programs as well.

Figure 5 shows one of the main steps in the tailoring process – this is actually step 8 of 12 steps (see Step 8 in Figure 6).

As you can see, most of the WI describes how the tailoring process is conducted (Steps 1-7 and 9-11 in Figure 6), how the tailoring decisions are stored in iPlan (Step 8), and how the tailoring is approved by the stakeholders and finally approved by the ultimate stakeholder (Step 12), the Engineering Process Group (EPG). So, a program cannot go wild and tailor out important portions of the CPA. Tailoring is a key concept in the CMMI model, and it is a critical task for programs using our CPA, and critical to survival for small programs using CPA. The model recognizes that many factors impact the process implementation on a particular program, for example:

- Program size.
- Program complexity.
- Contractual requirements.
- Customer deliverable requirements and format requirements.
- Specific measurements required by the organization and the customer.

Figure 5: *Main Step in Tailoring Work Instruction*[5]

8. Make Tailoring Decisions	**Requirement** Tailor the CPA Work Instruction Requirements (WIRs) for each WI. **Guidance** 1. For each work instruction provided by the iPlan tool, the tailoring team evaluates the work instruction requirements to determine the program tailoring decisions. WI Requirement decisions are the following: a. Accepted b. Rejected c. Modified The *Tailoring Decisions and Work Instruction tab* in the Tailoring Workspace of the iPlan tool provides visibility into the WIRs for each WI and allows the program to select or deselect specific WIRs and provides a utilization summary for each WI. The *Library Utilization* section of the *Work Instruction tab* is used to record the programs tailoring decisions with regard to a specific set of WIRs. Work Instruction Utilization Acceptance is automatically determined by iPlan based on the following: a. Accept – all WIRs accepted. b. Modified – one or more (but not all) WIRs rejected c. Reject – all WIRs rejected. 2. This process is repeated for all work instructions provided for this tailoring workspace. Note that the tailoring workspace may be different for each discipline and for the program level (multi-discipline). 3. The tailoring team consults with the EPG representative or iPlan SME to evaluate the effect on higher requirements when WIRs have not been accepted. Although tailoring is not to be considered something negative and programs should do what is smart for their business, the organization has an interest in understanding how and why programs are deviating from the organizational standards. Keeping with this notion, when a program tailors or rejects WIRs, the Notes field for that set of WIRs must be filled in to address 'why' the program is deviating from the organizational standard.

As far as our CPA tailoring, each of the WI requirements can be accepted (i.e., accepted as written), rejected (i.e., deleted from the work instruction) or modified (i.e., rewritten, such as SDP will be written to customer format versus CPA format). The Guidance for each of the CPA Requirements can be adjusted for the specifics of the program and contractual requirements, the output documents can be rejected or modified to meet the contractual requirements, and they can be adjusted for the program size. All tailoring is captured in the iPlan tool as modified WIs for the specific program. The iPlan tool provides a Program Tailoring Workspace – a virtual library for the program's approved process definition. The iPlan tool captures all the tailoring done to each of the CPA WIs, the EPG approved justification for all the tailoring, and the actual modified WIs for reference by program engineers during implementation. I must emphasize again: that all tailoring is approved by the EPG before the program's tailored process is implemented. For larger programs with extended schedules, the tailoring of the CPA process can be done incrementally by program phase so you do not have to do tailoring of the system test process if that process is not needed for several years. This process flexibility is expected in the CMMI and is an integral part of the RNCS CPA tailoring process. It is especially important for the small programs that have to survive among the big program processes, which leads us to small project tailoring.

Small Project Tailoring

One of the first steps, of course, is to understand what it means to be a small project. In the beginning of CPA, it was more a concept of what a small program consisted of, i.e., less than a year duration, small budget, small team, etc. Small programs are now formally defined as those with budgets between $500,000 and $4 million, less than a year in duration, and/or a team of eight or less engineers from all disciplines. We now have a definition of a small program which is helpful when we enter the tailoring process. The tailoring process is formalized during the start of the program. Now one of the major structural aspects of Raytheon's IPDS and reinforced in RNCS CPA is the program gating process. Formal gates are reviewed at specific engineering phases: program start-up (gate 5), system requirements

review (gate 6), preliminary design (gate 7), detailed design (gate 8), formal testing (gate 9), etc. As a matter of fact, we have 11 formal gates, four before contract award and seven gates after contract award. The important one for small project tailoring is our gate 5, known as program start-up. Every program right after contract award has 45 days to complete the program start-up process and review the overall program approach with the various senior management levels associated with the execution of the program. Key to this start-up process is the tailoring of the CPA WI to describe how each WI is going to be implemented on the program. As previously stated, in the proposal phase, several of the engineering plans may have been written in draft form or written as part of the proposal. After contract award, during the program start-up phase, the various engineering plans are either going to be updated from the drafts written in the proposal phase or written in the start-up phase. The engineering plans consist of the following:

- SEP.
- SDP.
- HDP.
- Stakeholder Involvement Plan (SIP).
- Measurement Analysis and Improvement Plan (MAIP).
- Integration, Verification and Validation Plan (IVVP).
- Quality Program Plan (QPP).
- WPMP and Work Product List (WPL).
- Several others.

As you can imagine, this can be an intensive period for the managers involved with the program. The small program tailoring of the CPA WIs is a heavy duty effort. Just think of trying to pare down 88 WIs in order to describe how you are going to implement the program using these tailored CPA WIs. The first group of small programs that pioneered their way through this tailoring process and tailoring justifications were reused as the various plans were adjusted and reused. Well, it did not take long to determine that an organizational process improvement was necessary to assist these small programs through the program start-up and tailoring processes: enter the Small Program Variant (SPV) Planning WI.

SPV Planning

The RNCS CPA Engineering Councils collaborated and generated a new WI to be used by small programs to do their program planning called the SPV Planning WI. It essentially provided pre-tailoring of the CPA WIs as a starting point. One of the major steps in the SPV planning was the creation of the Small Program Engineering Plan (SPEP) template. The SPEP template combined the major portions of the SEP, SDP, HDP, and IVVP into the single SPEP. Furthermore, additional planning activities associated with large programs were reduced for the small programs. For example, the amount of measurement data is reduced, such as staffing, since it is easy to keep track of three or four people. Typically, small programs deal with only one or maybe two of the engineering disciplines, so specific discipline WIs can be eliminated, for example: A software-only job can eliminate all the WIs associated with Hardware Preliminary Design, Hardware Detailed Design, Hardware Testing, etc. The details of stakeholder involvement can be reduced on small teams. Details such as formal weekly team meetings can be eliminated especially when the team is two to three people all working in the same cubicle. Formal reviews are reduced to three to four page review packages versus 35 to 40 page detailed, senior manager review packages. Small programs still have to deal with 78 WIs and the tailoring of those, but with the creation of the SPEP a significant work load was removed from the programs. There are, however, in the SPV Planning WI some of the other engineering plans, even for the small program, that have to be created and implemented such as the following: the Risk and Opportunity Management Plan, the MAIP, the WPMP, WPL, and the SIP. On my small programs, I have even been able to convince the EPG that some of these required standalones could be downscoped and incorporated into my program's single SPEP, thus reducing the coordination and sign-off cycle of multiple, individual plans; it was done using just my one SPEP.

> *"Key to this start-up process is the tailoring of the CPA WI to describe how each WI is going to be implemented on the program ... the engineering plans may have been written in draft form or written as part of the proposal."*

Other allowable tailoring is the combination of engineering phases. Typically, small programs are associated with extending the functionality of product lines where the preliminary design of the functionality is already known. So, small programs can tailor the conduct of two of the engineering gates reviews into one: the preliminary design and detailed design phases and the associated gates 7 and 8 (Preliminary Design Review [PDR] and Critical Design Review [CDR]). Even more typical, the system requirements are usually small changes to the product line and the System Functional Requirements Review (gate 6) can also be rolled into the combined PDR/CDR. This is all dependent upon the tailoring and permission given by the EPG during the EPG's tailoring approval meeting.

Implementation of Small Programs

With the SPV Planning WI, how does the implementation of small programs differ from the standard programs? To start, the number of WIs to tailor is reduced to 78 WI for SPV down from the 88 used in standard programs. The combined SPEP document combines several of the

Figure 6: *Steps at a Glance of the Tailoring Process*[6]

ID	Steps at a Glance	Responsible Role
1	Establish Tailoring Team	Program Engineer
2	Use CPA Tailoring Guidelines	Tailoring Team
3	Establish Tailoring Workspace	Tailoring SME
4	Execute Tailoring Kickoff	Program Engineer
5	Select Program Characteristics	Tailoring Team
6	Identify Stakeholders	Tailoring Team
7	Document Tailoring Stakeholders	Tailoring Team
8	Make Tailoring Decisions	Tailoring Team
9	Document Planning Strategy	Tailoring Team
10	Capture Tailoring Workspace	Program Engineer
11	Schedule Stakeholder Review	Program Engineer
12	Obtain Stakeholder Approval	Program Engineer, Tailoring Stakeholders

required plans and averages around 45-60 pages. The volume of the individual plans needed for standard programs collectively can be in the hundreds of pages. Measurements are key concepts in high maturity organizations. Regardless of the size of the program (standard, small, or micro) they need to collect the typical EVMS metrics needed to analyze the following: Schedule Performance Index (SPI) and Cost Performance Index (CPI). The difference, however, is the number of cost accounts that are tracked by small programs are around 30 to 40, where in standard programs the number of cost accounts can be in the hundreds of account numbers. Naturally, on small programs the number of data points collected is quite limited. Neverless, there are other measurements recommended in the small program tailoring guidelines in addition to CPI/SPI to manage these small programs. From the *Tailoring Guidelines:*

> The metrics that a [small] program collects is based on the characteristics of the program and customer requirements. The following is a recommended list of metrics that a small program should collect unless there is a valid reason to tailor out, such as:
> - CPI/SPI.
> - Defect Containment [which are collected through peer review/ inspection data and system trouble reporting data].
> - Requirements volatility.
> - Any (i.e. hardware, software, systems) applicable engineering productivity measures.[7]

The small program task managers indicate what measurements they use to manage their programs in their MAIP. The MAIP which identifies the program's measurements requires approval by the EPG. Other adjustments are allowed for small programs during the tailoring process, such as the following:

- Even though they started with 78 potential WIs, and since the disciplines such as hardware can be tailored out if there is no hardware development, the number of applicable WIs can get down to about 20 and those can be substantially tailored even further.
- Monthly review packages (35-40 slides) are reduced to four square charts (three to four slides).
- Process Support Team (PST) meetings where metrics analysis and process compliance checks are reduced both in the team size and the time to conduct the PST meetings.
- The 10 required plans can be combined with the SPEP, the implementation of the various discipline plans are not as detailed, and the associated enablers and templates are also reduced in size.

All the tailoring, even though much reduced, still requires a significant effort by the small program's management team. Therefore, the next step is for the EPGs and the senior level discipline councils to provide even more pre-approved tailoring for each of the WIs and the WIs associated enablers. Going one step further, the follow-on pre-tailoring activity will be to address micro-programs. These micro-programs are even smaller than the small programs I just discussed. The micro-programs typically have work effort content between 1,000 and 8,000 staff hours. Currently, there are a number of WIs being piloted by the RNCS organization that target these micro-programs. Yes, I know what you are thinking – what about the projects under 1,000 hours ... small projects among big trees; ever think about where toothpicks come from?

Summary

If we used the CMMI through Maturity Level 5 itself as a process description, it would be well over 600 pages. So when you establish your standard processes and actually write the organizational process description you can see that all the process-oriented documentation and implementation of these processes can become an insurmountable issue for small programs (e.g., small budget, small team, short schedule). The CMMI recognizes that tailoring is key to the successful implementation of the organizational processes for the variety of programs that organizations have in their business. It is up to the organization to provide the guidance, methodology, and the expectations of the small programs that are being executed in the same process environment as the big trees. An organization has to have an organizational-approved tailoring method for the small programs in order to survive and be successful, while still supporting the essence of the organizational processes. Tailoring is a life-saver for our small programs that also support our overall CMMI Maturity Level 5.◆

References

1. SEI. "Process Maturity Profile, CMMI V1.1 SCAMPI℠ v1.1." Class A Appraisal Results 2007 Mid-Year Update. Sept., 2007.
2. SEI-Carnegie Mellon University, Published Appraisal Ratings. June, 2007 <http://sas.sei.cmu.edu/pars/pars.aspx>.

Notes

1. RNCS Common Process Architecture Detailed Planning Work Instruction, #U0141CRP, dated 01/22/2007, pg 1.
2. RNCS Common Process Architecture Detailed Planning Work Instruction, #U0141CRP, dated 01/22/2007, pg 1.
3. RNCS Common Process Architecture Detailed Planning Work Instruction, #U0141CRP, dated 01/22/2007, pg 2.
4. RNCS Common Process Architecture Detailed Planning Work Instruction, #U0141CRP, dated 01/22/2007, pg 3.
5. RNCS Common Process Architecture Tailoring the CPA Process Work Instruction, #U0141CON, dated 01/12/2007, pg 6.
6. RNCS Common Process Architecture Tailoring the CPA Process Work Instruction, #U0141CON, dated 01/12/2007, pg 1.
7. CPA Tailoring Guidelines, Enabler, U0141H10, rev D, dated 10/05/2007, pg 11.

About the Author

Alan C. Jost (Lt. Col., U.S. Air Force, Ret.) is a senior software program manager in the Raytheon Northeast Software Engineering Center (SWEC), where he works multiple tasks including the Software Engineering Program Group Executive Committee for SWEC's CMM/CMMI Level 5 sustainment and as a process engineer on the AutoTrac III Air Traffic Control Product Line and DD(X) ExComms Software Cross Product Team. Jost joined Raytheon after 20 years in the Air Force. He has served in a variety of line management, process engineering, and software task management roles in SWEC.

Raytheon
MS 3-1-3914
1001 Boston Post RD
Marlborough, MA 01752
Phone: (508) 490-4282
Fax: (508) 490-1366
E-mail: alan_c_jost @raytheon.com

Field Guide to Provide Step-by-Step Examples for Improving Processes in Small Settings

Caroline Graettinger, Suzanne Garcia, and William Peterson
Software Engineering Institute

Christian Carmody
University of Pittsburgh Medical Center

M. Lynn Penn
Lockheed Martin Corporation

To help organizations in small settings pursue process improvement, the Software Engineering Institute (SEI) is inviting contributors to help develop a field guide with how-to guidance, examples, templates, checklists, and other information.

If you work for a small business, participate on a small team, belong to a small business unit, or work on small projects, then you work in a small setting and are likely familiar with the challenges of process improvement in these contexts.

Consider the following example from a small business: The Chief Executive Officer complains that she does not have enough cycles or resources to try out even one of several process improvement concepts that her various customers are asking for. She recognizes the risks of inaction, but the cost seems prohibitive, and she has never used consultants before. Her employees are all busy and many believe that *it's always worked fine the way it is*. She would like to have repeatable, predictable work processes that produce quality products and services and enable her company to stand out from the crowd. She knows she needs some kind of process improvement activity. But is her company ready? Is she ready? How can they become ready? What can they do themselves, and how can they be better consumers of process improvement products and services?

The SEI has heard stories like this from small settings around the world, and began to explore this arena in 2003 and 2004 with a pilot study in Huntsville, Alabama. The study resulted in new knowledge and ideas for how to accelerate implementation of one improvement methodology – Capability Maturity Model Integration (CMMI®) – in small businesses [1, 2]. This was followed by an insightful workshop in October 2005 involving researchers from around the world [3].

Based on this work and recommendations from the International Process Research Consortium, the SEI launched the Improving Processes in Small Settings (IPSS) project, in collaboration with University of Pittsburgh Medical Center (UPMC) and Lockheed Martin Corporation (LMCO). Why would UPMC and LMCO – whose employees number in the tens or hundreds of thousands – be interested in a project for small settings? Because, like many large organizations, they are amalgams of many small projects and business units, with myriad small business partners and suppliers.

The first IPSS project is the Field Guide for Improving Processes in Small Settings. The guide is not constructed like CMMI or any other process improvement models or frameworks; it is a collection of how-to guidance for process improvement in small settings, independent of the process model or standard used. We intend it to help fast-track the improvement effort and convey the scope of effort and skills involved at each step so that the small-setting practitioner can be a smarter consumer of process improvement products and services or be better at doing it themselves, whichever they choose.

The information in the field guide is organized under six competencies: (1) building and sustaining sponsorship and ownership; (2) developing and measuring realistic goals; (3) developing and sustaining a process improvement infrastructure; (4) defining and describing processes; (5) developing new or improved processes; and (6) determining improvement progress. Each competency comprises a set of activities that describe *what* to do to achieve that competency, and each activity comprises a set of tasks that explain *how* to do each activity.

Our plan for populating the field guide includes collecting real-world experiences from experts across the process community who can provide knowledge, techniques, examples, checklists, scripts, and other artifacts to help others succeed in small settings. The guidance will include step-by-step tasks for various situations and constraints of the small setting.

We welcome the involvement of small settings experts, citizens, and their stakeholders to help accelerate the development of the field guide. There are currently three ways to participate:

- Become a project affiliate and work directly on the guide at various stages.
- Become a project sponsor, which enables organizations to influence the priority order of the guide's development and content for their particular needs.
- Complete the brief survey we have created to collect information at <www.sei.cmu.edu/iprc/ipss.html>.

For more information on the field guide and the IPSS project, please visit <www.sei.cmu.edu/iprc/ipss.html>.◆

References

1. Chrissis, M., M. Konrad, and S. Shrum et al. "CMMI: Guidelines for Process Integration and Product Improvement v1.2." Boston: Addison-Wesley, 2006.
2. Garcia, S. Highlights from Piloting CMMI With Two Small Companies. Proc. of the First International Researcher's Workshop on Process Improvement in Small Settings. Pittsburgh: SEI, Carnegie Mellon University (CMU), 2006 <www.sei.cmu.edu/publications/documents/06.reports/ 06sr001.html>.
3. Garcia, S., Caroline Graettinger, and Keith Cost. "Proc. of the First International Researcher's Workshop on Process Improvement in Small Settings." Pittsburgh: SEI, CMU, 2006 <www.sei.cmu.edu/publications/documents/06.reports/06sr001.html>.

About the Authors

The authors are members of the IPSS project at the SEI and were part of the International Process Research Consortium that, from 2004-2007, brought together leaders from the international process community to explore strategic research directions in software and systems process. Caroline Graettinger, Suzanne Garcia, and William Peterson are senior members of the SEI CMU technical staff. Christian Carmody is Director of Process and Performance Improvement for UPMC. M. Lynn Penn is Director of Process Management at LMCO Integrated Systems and Global Services.

Carnegie Mellon
Software Engineering Institute
Phone: (412) 268-6109

CROSSTALK
The Journal of Defense Software Engineering

Get Your Free Subscription

Fill out and send us this form.

517 SMXS/MXDEA
6022 FIR AVE
BLDG 1238
HILL AFB, UT 84056-5820
FAX: (801) 777-8069 DSN: 777-8069
PHONE: (801) 775-5555 DSN: 775-5555

Or request online at www.stsc.hill.af.mil

NAME:_____

RANK/GRADE:_____

POSITION/TITLE:_____

ORGANIZATION:_____

ADDRESS:_____

_____

BASE/CITY:_____

STATE:_____ZIP:_____

PHONE:(____)_____

FAX:(____)_____

E-MAIL:_____

CHECK BOX(ES) TO REQUEST BACK ISSUES:

NOV2006 ☐ MANAGEMENT BASICS
DEC2006 ☐ REQUIREMENTS ENG.
JAN2007 ☐ PUBLISHER'S CHOICE
FEB2007 ☐ CMMI
MAR2007 ☐ SOFTWARE SECURITY
APR2007 ☐ AGILE DEVELOPMENT
MAY2007 ☐ SOFTWARE ACQUISITION
JUNE2007 ☐ COTS INTEGRATION
JULY2007 ☐ NET-CENTRICITY
AUG2007 ☐ STORIES OF CHANGE
SEPT2007 ☐ SERVICE-ORIENTED ARCH.
OCT2007 ☐ SYSTEMS ENGINEERING
NOV2007 ☐ WORKING AS A TEAM
DEC2007 ☐ SOFTWARE SUSTAINMENT
JAN2008 ☐ TRAINING AND EDUCATION

TO REQUEST BACK ISSUES ON TOPICS NOT LISTED ABOVE, PLEASE CONTACT <STSC.CUSTOMERSERVICE@HILL.AF.MIL>.

WEB SITES

Proceedings of the First International Research Workshop for Process Improvement in Small Settings, 2005

www.sei.cmu.edu/publications/documents/06.reports/06sr001.html

The first International Research Workshop for Process Improvement in Small Settings was held October 19-20, 2005 at the Software Engineering Institute in Pittsburgh, Pennsylvania. Attendees from Australia, Canada, Chile, China, Germany, Ireland, India, Japan, Malaysia, Mexico, Spain, and the United States discussed the challenges of process improvement in small and medium size enterprises, small organizations within large companies, and small projects. The presentations addressed starting and sustaining process improvement, qualitative and quantitative studies, and using Capability Maturity Model Integration (CMMI), Agile, Modelo de Procesos para la Industria de Software, International Organization for Standardization, Quality Function Deployment, and Team Software Process in small settings. The workshop also had working groups that discussed issues unique to small settings, such as regional support centers and process improvement on a shoestring. This report includes the papers from this workshop and presents conclusions and next steps for process improvement in small settings. This report also contains the workshop breakout session results.

Extreme Programming

www.extremeprogramming.org

Extreme Programming (XP) is a deliberate and disciplined approach to software development. About eight years old, it has already been proven at many companies of all different sizes and industries world wide. This web site gives an overall view of XP and presents numerous resources for learning more about the programming method.

Software Technology Support Center

www.stsc.hill.af.mil

In 1987, the U.S. Air Force selected Ogden Air Logistics Center (OO-ALC), Hill Air Force Base, Utah, to establish and operate its Software Technology Support Center (STSC). It was chartered to be the command focus for proactive application of software technology in weapon, command and control, intelligence and mission-critical systems. The STSC provides hands-on assistance in adopting effective technologies for software-intensive systems. We help organizations identify, evaluate, and adopt technologies that improve software product quality, production efficiency and predictability. We help others buy and build software and systems better. We use the term technology in its broadest sense to include processes, methods, techniques, and tools that enhance human capability. Our focus is on field-proven technologies that will benefit the Department of Defense mission.

Software Engineering Institute's Capability Maturity Model Integration Web Site

www.sei.cmu.edu/cmmi

The CMMI is a process improvement approach that provides organizations with the essential elements of effective processes. It can be used to guide process improvement across a project, a division, or an entire organization. CMMI helps integrate traditionally separate organizational functions, set process improvement goals and priorities, provide guidance for quality processes, and provide a point of reference for appraising current processes. This page points you to places where you can find more information about CMMI, and describes the worldwide adoption and benefits of CMMI.

Practical Software and Systems Measurement

www.psmsc.com

Practical Software and Systems Measurement (PSM) was developed to meet today's software and system technical and management challenges. It is an information-driven measurement process that addresses the unique technical and business goals of an organization. The guidance in PSM represents the best practices used by measurement professionals within the software and system acquisition and engineering communities.

Systems & Software Technology Conference

29 April – 2 May 2008 • LAS VEGAS, NEVADA

Technology: Tipping the Balance

Welcome to the 20th installment of the Systems and Software Technology Conference (SSTC). Just about 20 years ago, Tetris was becoming the computer game of choice. VGA Graphics were gaining in popularity. A company known by its initials, IBM, was delivering the PS/2 computer. It had a funny new pointing device - a mouse. And the first ever Software Technology Conference was held. From the beginning, SSTC has focused on technology relating to the Department of Defense. We begin another decade with this same focus.

Our theme will be ***"Technology: Tipping the Balance"***. The idea behind the theme is to explore new and needed technologies, as well as lessons learned, which tip the balance in the favor of our Defense Services, providing them an asymmetric advantage.

Submittal Topics Included:

- Assurance and Security
- Estimating and Measuring
- Lessons Learned
- New Concepts and Trends
- Policy and Standards
- Processes and Methods
- Professional Development
- Robust Engineering
- Systems: from Design to Delivery

Who Should Attend:

- Acquisition Professionals
- Program/Project Managers
- Programmers
- System Developers
- Systems Engineers
- Process Engineers
- Quality and Test Engineers

For Complete Conference & Trade Show Information

www.sstc-online.org

REGISTER TODAY!

CrossTalk's 20th Anniversary

You're Invited

CrossTalk, The Journal of Defense Software Engineering, is celebrating its 20th anniversary.

From its inception, CrossTalk's goal has been to inform and educate readers on software engineering processes, policies, and other technologies. We love hearing from our readers and friends on how we are doing. As a free journal, these comments are the lifeblood of our existence: They keep our staff focused and our sponsors motivated and pleased. If you find reading CrossTalk saves you time and money, helps improve your processes, has helped save your project, or makes your life easier, we want to know about it. The comments we receive will be considered for inclusion in our anniversary issue this coming August.

Please send your feedback to CrossTalk's Publisher, Beth Starrett at beth.starrett@hill.af.mil.

THANK YOU

Quoting it all the time to substantiate project plans and estimates.

THANK YOU

Articles on IDEF0, Upper CASE technologies, AI, CMM, CMMI, TSP/PSP, PM, and metrics helped me save 33.5 million dollars on government programs.

THANK YOU

Failed software development team: + SLOC and none of it worked. Between CrossTalk and our tech advisor, we shipped 580K + SLOC with more functionality than originally planned ...

Small Boats Among the Big Ships

You've probably heard of the U.S. Navy's famous Patrol Torpedo (PT) boats of World War II. Originally armed with four machine guns and four torpedoes, they were pound-for-pound the Navy's most heavily armed vessels. The 80-foot wooden warships served throughout the Pacific, in the Mediterranean, and even in the English Channel. Their most famous actions include evacuating General Douglas MacArthur from the Philippines and assisting John F. Kennedy's exploits when his PT109 was cut in half by a Japanese destroyer. By the end of World War II, the PT boats had racked up an impressive list of accomplishments [1].

Why bring up PT boats in CROSSTALK? Sure, the editors like stories framed with little-known bits of military history, but they're an excellent analogy for this issue's *Small Projects, Big Issues* theme.

At the war's inception, the PT boat retained the basic mission of all battleship-age torpedo boats: Use stealth and speed to sink a capital ship using torpedoes. Initially, the enclosed waters of the Philippine and Solomon Islands provided opportunities to continue their historical mission. But opportunities for surface combat would soon wane. Aircraft and radar made it much more difficult for the PTs to get close enough to use their powerful stings.

Despite this, the PTs found themselves even more in demand. Scouting, interdicting enemy barge traffic, performing reconnaissance, rescuing downed flyers, giving close shore convoy support, gathering intelligence, supporting ground operations, and many other missions that came the PTs' way. They soon bristled with new weapons: auto-cannon, mortars, and rockets. Some PTs abandoned their torpedoes entirely for more guns! Occasionally, though, they were still able to slam a torpedo into a major warship, as they did during the Battle of Leyte Gulf – *the largest naval battle in modern history* [2].

Small projects can learn important lessons from the small boats. First, the PTs were able to change their basic mission of hunting capital surface ships by adapting to fill an important niche in overall naval strategy. Originally designed for a role in the big ship navy, the PT boats found that circumstances dictated a very different one. They adapted well to their new role due in large part to the compactness of the boat and its crew. All teams must be able to adapt, but change comes more easily to smaller entities. This built-in flexibility allows small teams to operate with less stringent operating processes, often much less.

More importantly, consider how a PT boat differed from other U.S. Navy warships. Table 1 compares a PT boat to a heavy cruiser. Both are warships, but the differences are striking. Crew training cannot be over-emphasized. Losing even one man on a small crew could be devastating if another sailor couldn't take over. So every PT sailor had to be able to do almost any job on the boat, just like a small project software developer needs to be able to do any job on a project: requirements, design, coding, information and technology, documentation, configuration management, and even leadership.

For leadership, consider the PT boat skipper. He knows each sailor in his crew personally. His chain of command: one executive officer. Before a mission, he can muster the entire crew for a direct, verbal briefing. He knows the complete capabilities of the boat: weapons, engines, communications, and performance. From the cockpit, he can issue orders to any member of the crew verbally or via hand signal. He's confident his cross-trained crew can step in should a shipmate be disabled. The short, focused PT boat missions alleviate him from the more mundane aspects of captaincy. In battle, he operates alone or with other PT boats, all with the same *modus operandi*. Naval doctrine of the day was developed for the larger ships, but the realities of the PT's missions made it clear that they needed different methods, different tactics – *different processes* – to achieve that result. Aided by their compact nature, the PT crews readily established their own successful processes.

If your project calls for a PT boat, put one in place and run it accordingly. Using processes and procedures established for the normal behavior expected of large-scale projects can easily be counterproductive – or worse. Don't operate a PT boat like a heavy cruiser or expect it to act like one. If you do, you'll be sunk.

—Dan Knauer
TLA
A Three-Letter Acronym Corporation

References

1. Keating, Bern. "The Mosquito Fleet." Scholastic Book Services, USA: Dec. 1971.
2. "Battle of Leyte Gulf." Wikipedia <http://en.wikipedia.org/wiki/Battle_of_Leyte_Gulf>.

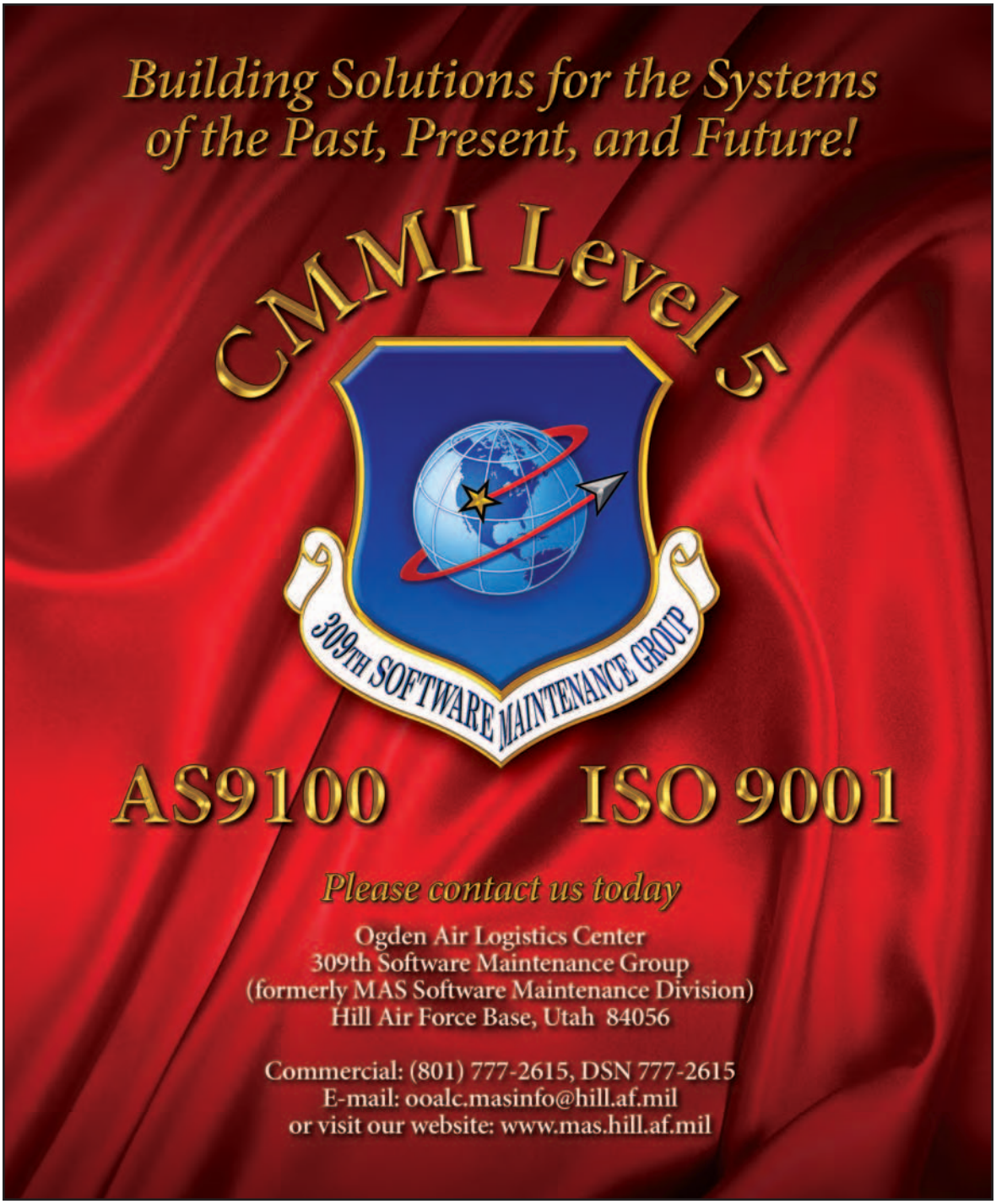
Additional Reading

The Internet provides several excellent sources of information on the PT boats including:

1. The Historical Naval Ships Organization <www.hnsa.org>.
2. PT Boats Inc <www.ptboats.org>.
3. John Drain's PT Boat Site <www.pt-boat.com>.

Table 1: *PT Boat Versus Navy Warship*

Aspect	PT Boat	Heavy Cruiser
Composition	Wood	Steel
Engines	Three Packard V-12 motors	Boiler-driven turbines
Fuel	Aviation gasoline	Oil
Armament ratio	One weapon per man	One weapon per 20 men
Transport to operating area	Carried aboard ship	Arrived under own power
Mission duration	Nightly patrols returning to same base	Multi-week cruises returning to varying ports
Crew training	Cross-trained in two disciplines; familiar with all tasks on boat	Single assignment plus battle station



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

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

CrossTalk is co-sponsored by the following organizations:

NAVAIR

Homeland Security