Carnegie Mellon
Software Engineering Institute

news @ sei
interactive

http://interactive.sei.cmu.edu

:

Editor's Message

The feature article titled "CERT's Function Extraction Project: Exploring Program Behavior for Security Analysis" in this issue of *news@sei interactive* describes an SEI exploratory research project related to the calculation of software behavior. The project's principal investigator, Rick Linger, invites the collaboration of interested individuals as a visiting scientist or affiliate. Over the past year, more than 200 people from government, industry, and academia have contributed to the improvement of software engineering practices by collaborating with the SEI.

Visiting scientists are full- or part-time temporary employees. Their work agreements are usually for three months to one year, but some have been extended for several years. They participate with technical staff on projects and are eligible to participate in SEI and Carnegie Mellon University activities. They offer a unique skill set, perspective, and practical experience that help in shaping the direction of the work within their respective programs. Information about visiting scientist positions is available from SEI Human Resources at hr-jobs@sei.cmu.edu.

In the Affiliate Program, participating organizations sponsor a qualified member of their technical staff to work with SEI staff members to identify, develop, and demonstrate improved practices for specific technology domains. Affiliates work with the SEI for about a year and for a certain percentage of their time. They have the opportunity to participate in public courses and workshops at minimal cost to their sponsoring organizations. Several affiliate opportunities are currently listed on the Affiliate Program Web site. For more information about the program, contact the program coordinator at affiliate@sei.cmu.edu.

Although we don't often report on exploratory research projects such as the Function Extraction Project in *news@sei interactive*, the SEI has several such projects underway all the time. Each year, the SEI undertakes several independent research and development projects. These projects are deemed to have the potential to contribute significantly to the maturation and/or transition of software engineering practices. A report published in October 2003, *SEI Independent Research and Development Projects* (CMU/SEI-2003-TR-019), describes the results of the Fiscal Year 2003 projects: Acquisition Simulation, Architectural Design Assistant, A Model-Based Reference Architecture for Mobile Robotics Systems, Securing Wireless Devices, Sustainment, and System of Systems Interoperability.

We hope you find something in this issue of *news@sei interactive* that helps you explore some new possibilities in your own work. Let us know—we welcome your feedback.

Pamela Curtis
Editor in Chief

:

Columns

The Architect

Integrating Analysis and Design Methods for the Software Life Cycle

MARIO R. BARBACCI

In the last issue of The Architect, “Rethinking the Software Life Cycle” (Volume 6, Number 3, Third Quarter 2003), the authors described architecture-centric analysis and design methods that have been emerging for a number of years. A number of these methods have been described in previously in The Architect. The methods share a set of common characteristics and activities that suggest the methods could be used in combination. In some cases activities are redundant; if this duplication is eliminated, using the methods in sequence can save cost and time. This column continues the theme and illustrates examples of combinations of the methods and when the combinations could be applied during the life cycle.

Most of these methods have a precise description of the documents and process required and usually the process is fairly rigid, not allowing for changes or deviations. We are now conducting an integration study, looking at how the various methods could be integrated and at the benefits of the integration. There are several technical reports to be published soon. This column is an invitation to the readers to send comments and to check the SEI Web site[1] at for new publications.

The Methods

Quality Attribute Workshop (QAW) [1] collects and organizes software quality attribute requirements. The QAW collects, prioritizes, and refines scenarios that can be used to test if the architecture will meet the requirements. The analysis proper is not part of the QAW process but can be performed as a step in some of the other methods. The inputs, outputs, and activities in QAW are illustrated in Figure 1.

1. http://www.sei.cmu.edu/ata/ata_eval.html

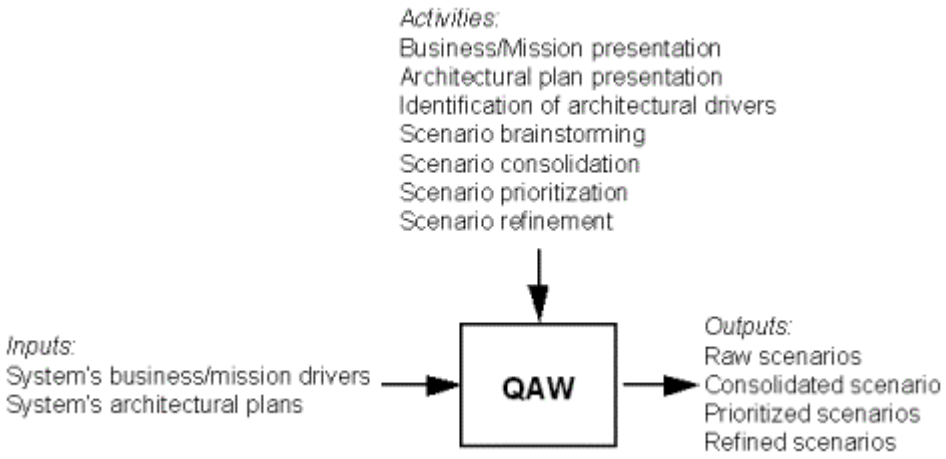

Figure 1: QAW inputs, outputs, and activities

Attribute-Driven Design (ADD) [2] defines a design process on the quality attributes the software must fulfill. ADD documents a software architecture in a number of views and depends on understanding the system's constrains and requirements, represented as scenarios. The ADD inputs, outputs and activities are illustrated in Figure 2.

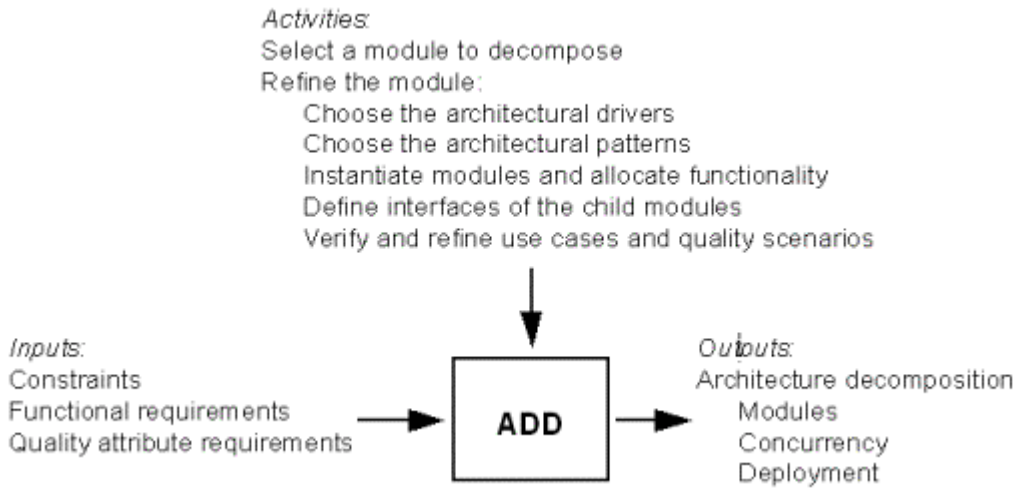

Figure 2: ADD inputs, outputs, and activities

Attribute Tradeoff Analysis Method (ATAM) [3] illuminates the consequences of architectural decisions with respect to quality attribute requirements. These consequences are illustrated in a set of risks, sensitivities, and tradeoffs. The ATAM inputs, outputs, and activities are illustrated in Figure 3.

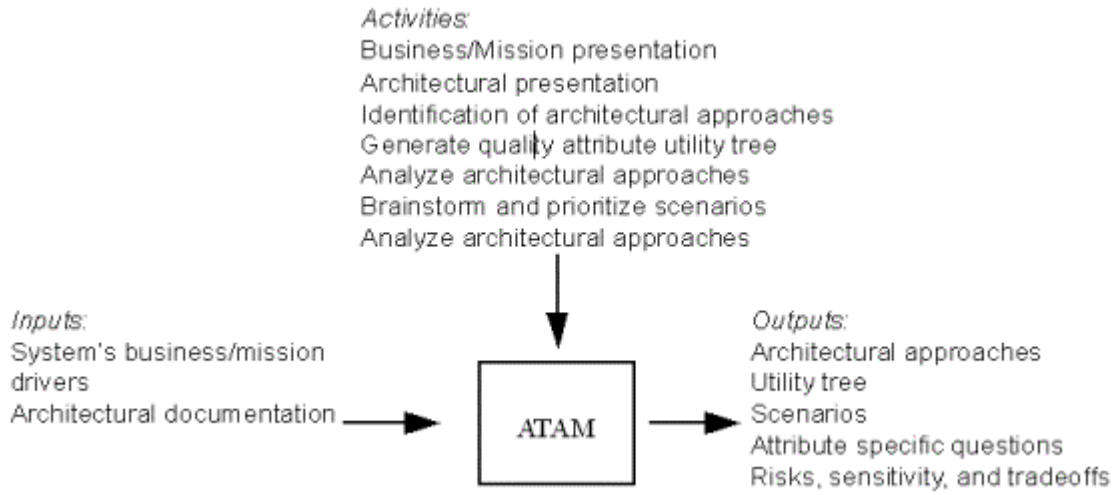

Figure 3: ATAM inputs, outputs, and activities

Active Reviews for Intermediate Designs (ARID) [4] blends active design reviews with ATAM into a technique for investigating designs that are practically complete. Like ATAM, ARID engages the stakeholders to create a set of scenarios that are used to test whether the design can be used by the software engineers who must work on it. The ARID inputs, outputs, and activities are illustrated in Figure 4.

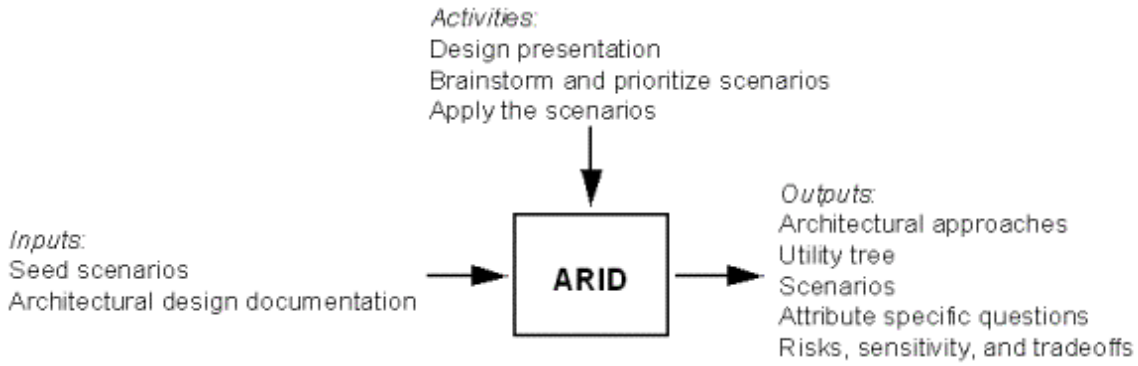

Figure 4: ARID inputs, outputs, and activities

Cost-Benefit Analysis Method (CBAM) [5] facilitates architecture-based economic analysis of software-intensive systems. This method helps the system stakeholders to choose among architectural alternatives during the design or maintenance phases. The CBAM inputs, outputs, and activities are illustrated in Figure 5.

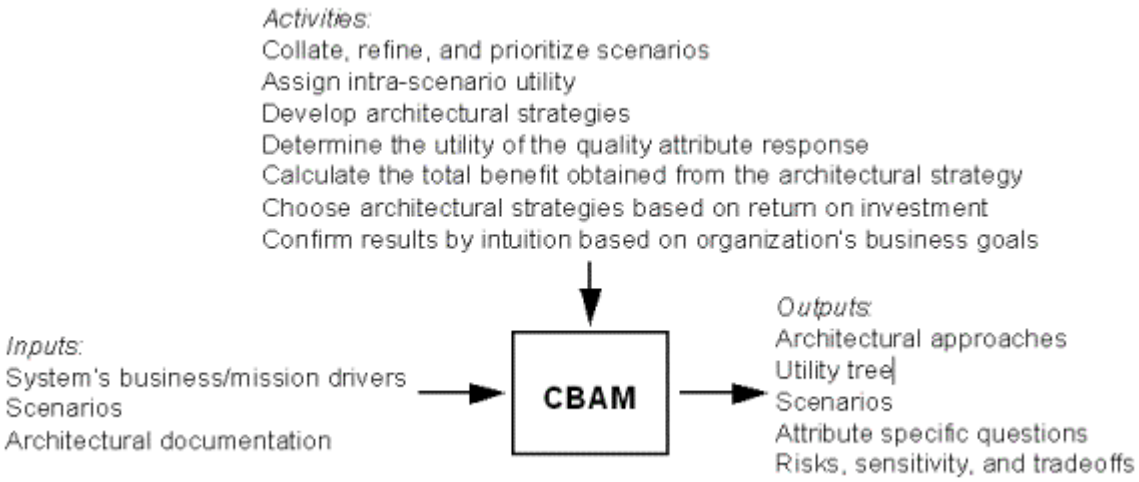

Figure 5: CBAM inputs, outputs, and activities

Combinations of Methods

Depending on the stage of the life-cycle, the methods can be combined by taking the results of a method and using them as input to another method, perhaps eliminating redundant activities. Table 1 shows the methods and activities, and notes which artifacts are inputs to the method, outputs from the method, or both. Figure 6 shows a sequence of applications of the methods.

Table 1:Methods and Life-Cycle Stages

Life-Cycle Stage	QAW	ADD	ATAM	CBAM	ARID
Business needs and constraints	Input	Input	Input	Input	
Requirements	Input; output	Input	Input; output	Input; output	
Architecture design		Output	Input; output	Input; output	Input
Detailed design					Input; output
Implementation					
Testing					
Deployment					
Maintenance				Input; output	

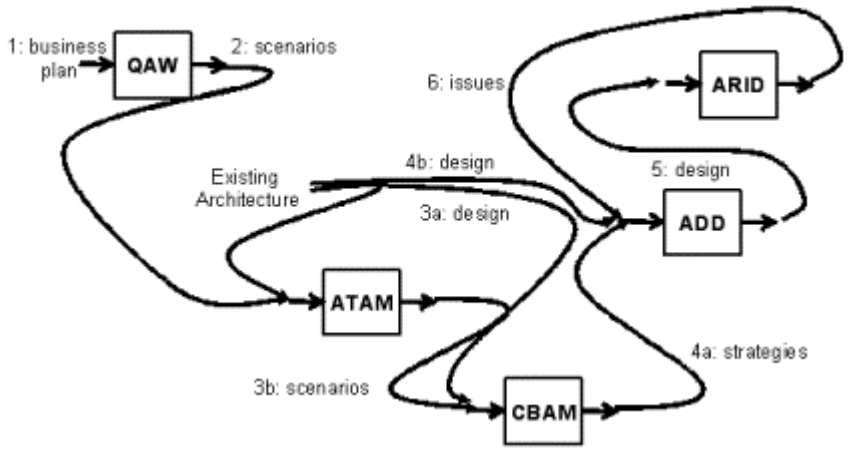

Figure 6: Combination of Methods

References

1. Barbacci, M.; Ellison, R.; Lattanze, A.; Stafford, J.; Weinstock, C.; & Wood, W. *Quality Attribute Workshops (QAWs), Third Edition* (CMU/SEI-2003-TR-016). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. <http://www.sei.cmu.edu/publications/documents/03.reports/03tr016.html>.

2. Bachmann, F.; Bass, L.; Chastek, G.; Donohoe, P.; & Peruzzi, F. *The Architecture Based Design Method* (CMU/SEI-2000-TR-001, ADA37581). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000. <http://www.sei.cmu.edu/publications/documents/00.reports/00tr001.html>.

3. Kazman, R.; Klein, M.; Clements, R. *ATAM: A Method for Architecture Evaluation* (CMU/SEI-2000-TR-004, ADA382629). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000. <http://www.sei.cmu.edu/publications/documents/00.reports/00tr004.html>.

4. Clements, P. *Active Reviews for Intermediate Designs* (CMU/SEI-2000-TN-009, ADA383775) Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000. <http://www.sei.cmu.edu/publications/documents/00.reports/00tn009.html>.

5. Kazman, R.; Asundi, J.; & Klein, M. *Making Architecture Design Decisions: An Economic Approach* (CMU/SEI-2002-TR-035, ADA408740). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002. <http://www.sei.cmu.edu/publications/documents/02.reports/02tr035.html>

About the Author

Mario Barbacci is a Senior Member of the staff at the Software Engineering Institute (SEI) at Carnegie Mellon University. He was one of the founders of the SEI, where he has served in several technical and managerial positions, including Project Leader (Distributed Systems), Program Director (Real-time Distributed Systems, Product Attribute Engineering), and Associate Director (Technology Exploration Department). Prior to joining the SEI, he was a member of the faculty in the School of Computer Science at Carnegie Mellon University.

His current research interests are in the areas of software architecture and distributed systems. He has written numerous books, articles, and technical reports and has contributed to books and encyclopedias on subjects of technical interest.

Barbacci is a Fellow of the Institute of Electrical and Electronic Engineers (IEEE) and the IEEE Computer Society, a member of the Association for Computing Machinery (ACM), and a member of Sigma Xi. He was the founding chairman of the International Federation for Information Processing (IFIP) Working Group 10.2 (Computer Descriptions and Tools) and has served as chair of the Joint IEEE Computer Society/ACM Steering Committee for the Establishment of Software Engineering as a Profession (1993-1995), President of the IEEE Computer Society (1996), and IEEE Division V Director (1998-1999).

Barbacci is the recipient of several IEEE Computer Society Outstanding Contribution Certificates, the ACM Recognition of Service Award, and the IFIP Silver Core Award. He received bachelor's and engineer's degrees in electrical engineering from the Universidad Nacional de Ingenieria, Lima, Peru, and a doctorate in computer science from Carnegie Mellon.

CMMI in Focus

CMMI—Continuously Improving!

MIKE PHILLIPS

My last column focused on adoption progress, and mentioned what we could see coming. Recent activity and the importance of both of these areas has merited a further update in this issue.

How Is Adoption Growing?

We mentioned last quarter that course attendance for the Introduction to CMMI[1] (Capability Maturity Model Integration) class had reached a total of 8,837 people. Thanks to the increasing capacity provided by our licensed transition partners[2] offering the Introduction to CMMI course, over 10,000 people have attended this course as of the end of the year.

Another measure of interest is the number of times our CMMI Web[3] pages on the SEI Web site are accessed. During the summer of 2003, the number of visits hovered at 800,000 to 900,000 per month. In September, we recorded over a million visits for information on CMMI, and the number continues to grow.

Similarly, the mix of appraisals[4] conducted continues to shift from SW-CMM to CMMI. The percentage of appraisals using a CMMI model has grown from 13% in 2002 to about 19% as of early December 2003. We have also had a number of requests to approve the use of the Standard CMMI Appraisal Method for Process Improvement (SCAMPI) method with the SW-CMM. These requests have been granted on a case-by-case basis whenever we conclude that such an approach will assist the organization's upgrade from SW-CMM to CMMI.

Another measure of adoption is the number and types of CMMI-related presentations at conferences. The CMMI Technology Conference and User Group[5] that was held in Denver November 14-17 had a wealth of valuable presentations, which are available on the National Defense Industrial Association (NDIA) Web site.[6] They describe ways that organizations have increased the rate of their CMMI adoption and implementation. Of particular interest was the use of Six Sigma and ISO standards with CMMI. Those who used Six Sigma and CMMI found them

1. http://www.sei.cmu.edu/products/courses/courses.html
2. http://www.sei.cmu.edu/collaborating/partners/trans.partners.html
3. http://www.sei.cmu.edu/cmmi
4. http://www.sei.cmu.edu/cmmi/appraisals/appraisals.html
5. http://www.sei.cmu.edu/about/whatsnew/whatsnew.html#cmmi-techconf-ug
6. http://www.dtic.mil/ndia/2003CMMI/2003CMMI.html

to be very compatible. Others who used ISO standards with CMMI, including ISO 9000 as well as standards for safety and security, found the potential for a growing synergy.

At the SEPG Conference[1] planned for March 2004 in Orlando, Florida, two tracks on each of the four days will be devoted to CMMI. Despite this increase in the number of slots devoted to CMMI-related presentations, so many CMMI-related presentations were submitted that only about one of every six CMMI-related submissions could be accepted. Even after giving substantial weight to the views of the conference's review committee, which included many Software Process Improvement Network (SPIN) members, I found it difficult, as the final arbiter of the CMMI tracks, to select the best of so many excellent submissions. Unfortunately, many excellent and timely presentations will not be heard because they could not be accepted.

How Is the Transition Partner Program Working?

Two weeks before the CMMI Technology Conference and User Group in Denver, the SEI hosted a workshop in Washington, DC for all SCAMPI lead appraisers and Introduction to CMMI instructors. About 130 people, or about one third of the authorized appraisers and instructors, joined us there. The three-day workshop allowed various working groups to help us continuously improve the product suite and to evaluate the workshop as a way to increase the competencies of our transition partners in the future.

Of particular significance at this workshop was the work done on development of a code of conduct designed to guide all of the professionals in the process improvement field who are in some way authorized by the SEI. These professionals include those who work for the SEI and those who work for SEI-authorized transition partner organizations.

What Is the 90-Day Review Period for Updating the Product Suite?

When we launched CMMI Version 1.1 in early 2001, we promised to maintain the stability of that version for at least three years. We are keeping our promise, to help organizations more easily plan their adoption of CMMI. We receive occasional change requests for all CMMI products and have regularly updated an errata sheet for each CMMI model and for the key appraisal documents (i.e., *SCAMPI Method Definition Document* and *Appraisal Requirements for CMMI)*. These errata sheets correct the obvious errors found since the release of Version 1.1. However, some change requests have contained requests that CMMI model best practices be extended to include coverage of additional disciplines or a greater breadth of the product life cycle.

Some have requested that CMMI best practices be expanded to cover areas such as hardware engineering and safety and security assurance. Others have requested that CMMI best practices be

1. http://www.sei.cmu.edu/sepg

expanded to cover more of the product life cycle to include manufacturing, operations, and disposal.

To begin considering when and which improvements should be made to the CMMI Product Suite, a 90-day comment period, open to current users of the product suite, was announced in September. The recommendations received from those using the model today will help guide the plans for corrections and improvements tomorrow. In addition to the input received during the 90-day comment period, we will also consider the change requests submitted since release of the V1.1, as well as comments received as part of workshops such as those conducted as part of the CMMI Interpretive Guidance project[1] and the Transition Partner Workshop.[2]

What Happens Next?

The SEI will analyze all of the change requests received, including those that were received before the Version 1.1 update that were deferred and those received before the 90-day review. A summary of this analysis will be submitted to the CMMI Steering Group[3]. The Steering Group will then determine the guidelines for product suite change, as it did for Version 1.1, as well as a strategy to guide both Version 1.2 and future update mechanisms. Our overall timeline for change suggests that the remainder of Fiscal Year 2004 will be spent planning CMMI Product Suite improvements. Fiscal Year 2005 (i.e., October 2004 through September 2005) will be spent creating draft versions of CMMI products and piloting them. Version 1.2 CMMI products will be finalized for release in Fiscal Year 2006. This timing allows for full consideration of needed changes and maintains our commitment to stability.

What If I Submit a Change Request Now That the Review Period Is Over?

The 90-day review period was established to stimulate a focused review for product suite improvement in organizations using CMMI, and will help us prepare the improvement strategy for Version 1.2. Change requests are always welcome.

We have found that many changes to the training material can be made without affecting the rest of the product suite. There is not a specific deadline that would preclude consideration of change requests. In fact, we expect change requests as part of the piloting activities that will take place in Fiscal Year 2005. Of course, the later a change request is received, the less likely it is that it will affect Version 1.2 elements. So it is worthwhile to submit your change requests as soon as you can. For more information about submitting a change request, see “Submitting CMMI Change Requests and Comments.”[4]

1. http://www.sei.cmu.edu/publications/documents/03.reports/03sr007.html
2. http://www.sei.cmu.edu/products/events/appraisers-instructors-workshop/hart-white-maher
3. http://www.sei.cmu.edu/cmmi/background/steering-group.html

What Happens to Change Requests?

When we are ready to process change requests, we gather those that are similar into a change package. This package represents the team's best fusion of the ideas affecting specific parts of the models, appraisal method, or training. The change package is submitted to the CMMI Configuration Control Board[1], which has the responsibility for determining whether the change is justified and sufficient to correct or improve the product suite. Not every change package is approved. Those changes that are approved will be the basis for the preparation of an improvement package in which actual changes to the elements of the product suite are made.

Should I delay investing in CMMI until Version 1.2 is released?

The drafting and piloting of proposed changes will take us well into 2005 at the earliest. We expect that the scope of change for Version 1.2 will be narrow. We are committed to ensuring that these changes will align with the existing product suite so that no repetition of training and deployment efforts such as those associated with V1.1 and V1.0 will be necessary. Thus, a decision to delay your investment in CMMI until late 2005 or 2006 may put your organization behind its peers and competitors and would delay any benefits gained from CMMI adoption. Further, such a delay would not provide any investment savings.

Are There Any New SEI Publications About CMMI Available?

Four new documents related to CMMI were recently published by the SEI, and one is in progress. Descriptions of each follow.

Interpreting Capability Maturity Model Integration (CMMI) for Service Organizations—A Systems Engineering and Integration Services Example[2]
We have made available a technical note from a service-focused organization that is part of a large defense systems integration company. The report provides an example of how CMMI best practices can be interpreted for organizations that primarily provide services.

Demonstrating the Impact and Benefits of CMMI: An Update and Preliminary Results[3]
One of the most popular reports from the SEI, produced in 1994, summarized some early SW-CMM adoption successes from a number of organizations that agreed to be part of the study. This report presents selected results from 12 case studies of CMMI-based process improvement drawn from 11 organizations. While still limited, the case studies provide credible evidence that CMMI-

4. http://www.sei.cmu.edu/cmmi/models/change-requests.html

1. http://www.sei.cmu.edu/cmmi/background/config-control-bd.html

2. http://www.sei.cmu.edu/publications/documents/03.reports/03tn005.html

3. http://www.sei.cmu.edu/publications/documents/03.reports/03sr009.html

based process improvement can help organizations achieve better project performance and produce higher quality products. One section of the study reports on recent successes that began as SW-CMM efforts but are continuing as CMMI-based improvements.

CMMI Interpretive Guidance Project: Preliminary Report[1]
The Interpretive Guidance project was formed to collect information about how CMMI is being used by software, information technology (IT), and information systems (IS) organizations. The project collected data using a variety of channels and sources within the software and IT/IS communities. This report provides some early insight into the adoption of CMMI by the organizations that chose to participate.

Some highlights of this report include the following:

- When asked "Has your organization made a decision about adopting CMMI?" 48% of the respondents stated that adoption was in progress, 15% stated that CMMI was well institutionalized throughout their organizations, 10% stated that their organizations chose not to adopt CMMI, and 23% said the decision to adopt had not yet been made.
- When asked if, in their opinion, CMMI is adequate for guiding process improvement, over 77% of respondents agreed or strongly agreed.
- When asked if including both systems engineering and software in a single model has been a help for them, nearly 65% agreed or strongly agreed.

Interpreting Capability Maturity Model Integration (CMMI) for COTS-Based Systems[2]
Although commercial off-the-shelf (COTS)-based systems are covered in CMMI models, the authors of this report show how CMMI can be interpreted to enable those developing COTS-based systems to follow CMMI best practices more easily. The report describes what makes COTS-based systems different from systems that do not use COTS products.

CMMI model best practices cover the selection of COTS products and managing vendor relationships in the supplier sourcing discipline. This report interprets the best practices in other areas of the models (i.e., systems engineering, software engineering, and integrated product and process development) that can be affected by the use of COTS products.

CMMI Module for Acquisition (in progress)
One of the disciplines considered for eventual inclusion in the CMMI Framework is acquisition. In December 2000, incorporating this discipline was explored by releasing draft process areas in CMMI Version 1.02d. This draft later evolved into the supplier sourcing (SS) addition to Version

1. http://www.sei.cmu.edu/publications/documents/03.reports/03sr007.html
2. http://www.sei.cmu.edu/publications/documents/03.reports/03tr022.html

1.1. Supplier sourcing practices are contained in a single process area, *Integrated Supplier Management*, and informative amplifications of various practices in other process areas. This addition of a discipline increased attention on analyzing and selecting suppliers and improving customer-supplier interactions. In spite of this step forward for outsourcing, we determined that further acquisition practices would be needed in the future to help the government's acquisition organizations.

This report is designed to provide broader coverage of acquisition within the CMMI Framework and meet the needs of government acquirers. Significant news not included in my last column is that source material from the Federal Aviation Administration's Integrated Capability Maturity Model (iCMM) is being included to move the community toward a single integrated model for both government and industry. We intend to publish this report in early 2004. This report will provide material that can be reviewed, piloted, and tested before it is considered for inclusion in the CMMI Framework. As I mentioned before, this approach allows us to maintain the stability of the Version 1.1 Product Suite while we expand the communities that can benefit from CMMI.

What About Coverage For Safety And Security? (Coming)

A team jointly sponsored by the DoD and the FAA has made significant progress in identifying the key practices that need to be emphasized to ensure effective development of safety-critical and security-critical systems. An approach for incorporating these practices into the CMMI Framework was presented at the CMMI Technology Conference and User Group by the co-sponsors from the FAA and DoD. The team will be gathering more data before preparing a technical note. If you would like to participate in reviewing the material, please contact Matt Ashford at the SEI (ashford@sei.cmu.edu).

About the Author

Mike Phillips is the Director of Special Projects at the SEI, a position created to lead the Capability Maturity Model® Integration (CMMI®) project for the SEI. He was previously responsible for transition-enabling activities at the SEI.

Prior to his retirement as a colonel from the Air Force, he managed the $36B development program for the B-2 in the B-2 SPO and commanded the 4950th Test Wing at Wright-Patterson AFB, OH. In addition to his bachelor's degree in astronautical engineering from the Air Force Academy, Phillips has masters degrees in nuclear engineering from Georgia Tech, in systems management from the University of Southern California, and in international affairs from Salve Regina College and the Naval War College.

Security Matters

There IS an Intruder in My Computer—What Now?

LAWRENCE R. ROGERS

The day's chores are done and you're ready to sit down at your home computer to do a little recreational web surfing and to catch up with your online friends. As you log in, you notice that your modem's transmit and receive lights are on almost all the time. You soon discover that almost everything is running slower and slower with each new program you start. And some of those programs don't seem to be working as they once did. What's going on here?

Chances are your home computer system has suffered a break-in. And the changes made by the intruders probably won't go away by themselves – you need to fix your computer. So much for a relaxing evening!

Before you begin to tackle the task of fixing your broken computer, you first need to answer some questions. The answers will help to guide the repair process.

1. What changed? Intruders broke into your computer and probably changed it in some way. You need to figure out what they changed so that you can undo those changes.

2. How did they get in? Intruders took advantage of one or more weaknesses – *vulnerabilities* – in your computer. You need to figure out how they got in so that you can fix those vulnerabilities. It's important to know how they got in because if you don't, they may come back.

3. What do I need to change? Intruders may have been able to take advantage of vulnerabilities because of the way your computer is set up. You may need to make changes to thwart another attack. Examples are installing patches, installing and using a virus checker, and installing and using software and hardware firewalls.

To help you to think about how to answer these questions, imagine that your house was broken into instead of your computer. You're probably a lot more familiar with your house, so let's use this analogy to apply what you already know to the task of repairing your computer.

1. What changed? To answer the *What changed?* question about your house, you and those you live with have a pretty good idea of what it looked like before the break-in. Since you already know this, you can more easily figure out what changed.

With your computer, it's a lot harder to know what it looked like before the break-in, that is, what files and folders were on your hard disk and what they contained. And your computer is much more sensitive to the locations of files and folders and their contents than is your house and its contents. For example, if your television were stolen, you could replace it with another television.

But you don't need to replace it with exactly the same television, and you don't need to put it in exactly the same place.

In contrast, with your computer you do have to replace moved, deleted, and changed files and folders with exactly what was there before, in most cases, and put them in the same place. Anything less might prevent your computer from working correctly.

And it's not just the files and folders you've created. Your computer came with many files and folders that are part of the operating system (Windows® for example) and its applications (Microsoft® Word, Excel, Outlook, and others that you may have added, such as Intuit® Quicken). It is these files more so than your personal files and folders that must be in the same location and have the same content as before the break-in.

So, what did those files and folders look like before the break-in, and where were they? Chances are you're not really sure. Not only that, it's hard to tell unless you've taken steps to record the kind of information that can help you decide what changed.

To help you keep track of the files and folders on your computer and their content, you need to purchase or download a free version of a program called an integrity monitor. An integrity monitor checks files and folders to see what's changed since they were last inspected. These programs use advanced mathematics so that even the smallest of changes can be detected. You run an integrity monitor when you first set up your computer and then run it periodically to see what changed since the last time you checked. Sounds easy, doesn't it?

Unfortunately, it's not as easy as it sounds. The challenge is to know what changes to files and folders can be expected as part of the routine operation of your computer. Unexpected changes are likely to be the result of something else, an intruder's activities for example.

How do you know what changes to expect? The vendors could tell you but they don't, so you'll have to figure that out for yourself. One way to do this is to run the integrity monitor once to determine what files and folders are there and what they contain. This process is called *baselining*. After you've baselined your computer, use it for a while and then run the monitor again. The changes you see should be expected. Over time, you'll learn what's okay and what's not. And yes, it is a time-consuming and tedious process. But if you want to know what's changed, that's what it takes.

Now, what do you do if you haven't baselined your computer, so you can't figure out what the intruder changed? The safest thing to do is to start all over again by formatting your hard drives, reinstalling your system and applications, and restoring important files from your backups. (You do have backups, don't you?) Otherwise, you will be relying on files and folders that an intruder may have changed that are still on your hard disk. It's a risk to use these files but a risk you might decide to accept. Nonetheless, when you don't know for sure what files were changed, it's safest to start all over again.

2. How did they get in? Back at your house, there may be obvious signs such as a broken window or a pried-open door. However, if the intruder used an unlocked door and then locked it on the way out, there may be no break-in signs. The point is sometimes it's easy to figure out and other times it's nearly impossible.

Figuring how the intruders got into your computer presents the same problems. If intruders took advantage of a vulnerability in a program, there may no signs that that's what they did. But if they sent an email message with a virus attached and text that belittles its reader for poor email security, then it's more obvious how the break-in occurred. This type of computer intrusion analysis—called *computer forensics*—can be more art than science at times. Clues may be hard to come by.

But a little detective work might reveal what happened to your computer. For example, if you use an integrity monitor, you might be able to determine how the intruders got in based on the files they changed. Since viruses are a popular method intruders use to break into computer systems, you may be able to relate the set of files changed on your computer to a virus the intruders used. Visit the anti-virus web sites to get information about viruses and the files they change. This may give you the clues you need to figure out how the intruders got in.

There may be other clues lying around. For example, your computer might run slower and slower, as mentioned in the first paragraph. This could be an indication that one or more programs are running constantly, perhaps sending traffic to other computers. Review the programs that are running on your computer and see which are using the most CPU time. Use this information as you used the "files changed" information from your integrity monitor to try to learn how the intruders broke in to your computer.

3. What do I need to change? Answering the What do I need to change? question about your house may be easy. You may keep a key to the front door under the flower pot on your front porch. A passerby who happens to see that there is a key there could use it to enter your house. This is a case where you need to change the way you set up your house to improve its security.

Now back to your computer. If you can figure out how the intruders got in, you can more easily decide what you need to change to try to keep them out. For example, if they took advantage of a vulnerability and you don't routinely install patches that would have fixed the vulnerability, then you need to start installing patches routinely. Or, if you use your home computer to read email and you aren't running a virus scanner, you should install anti-virus software and use it, especially when working with email. For a list of tasks you should routinely do to your home computer, read *Home Computer Security (*http://www.cert.org/homeusers/HomeComputerSecurity/).

The message is that you might need to change how your computer is set up so that it is more resistant to intruders.

Many home computers have been broken into, perhaps even yours. Knowing what to do after the break-in requires some advanced planning to reduce your repair effort and can greatly improve the chances that you've completely rid your computer of the intruder. It's time well spent.

About the Author

Lawrence R. Rogers is a senior member of the technical staff in the Networked Systems Survivability Program at the Software Engineering Institute (SEI). The CERT Coordination Center is a part of this program. Rogers's primary focus is analyzing system and network vulnerabilities and helping to transition security technology into production use. His professional interests are in the areas of the administering systems in a secure fashion and software tools and techniques for creating new systems being deployed on the Internet. Rogers also works as a trainer of system administrators, authoring and delivering courseware. Before joining the SEI, Rogers worked for 10 years at Princeton University. Rogers co-authored the Advanced Programmer's Guide to UNIX Systems V with Rebecca Thomas and Jean Yates. He received a BS in systems analysis from Miami University in 1976 and an MA in computer engineering in 1978 from Case Western Reserve University.

This article is adapted from Task 3 in *Home Computer Security*, which can be found at http://www.fedcirc.gov/library/documents/homeusers/index.html and http://www.cert.org/homeusers/HomeComputerSecurity/. This work was funded by the General Services Agency of the U.S. Government.

Software Product Lines

It Takes Two

PAUL CLEMENTS

People considering the move to a product line strategy usually get around to asking some form of the question, “So how much is this going to cost us in the short term?” A time-based version of the question is, “How many days/months/years will it take for my up-front investment to be recouped by cost savings?” As you can imagine, it’s a difficult question to answer, because of the enormous variations among organizations, products, and markets. The answer is an unsatisfying “It depends.”

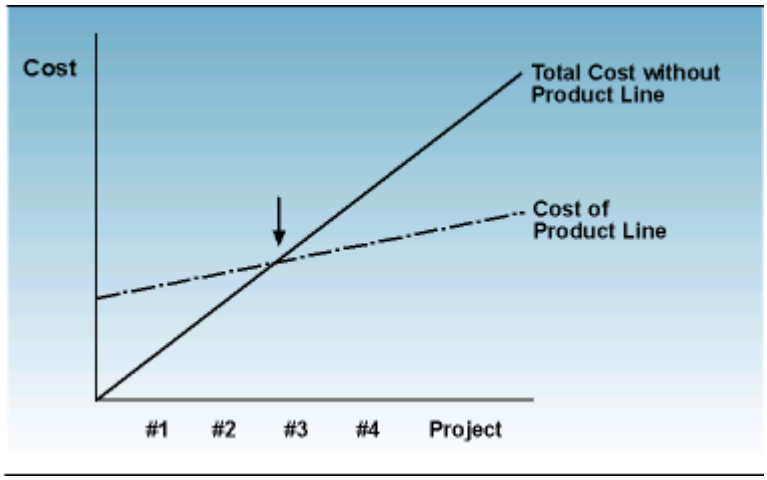

Figure 1: Software Product Line Payoff Point

But that is only if the questioner insists on an answer given in dollars or days. It so happens that there is a very satisfying way to re-frame the question that normalizes the answer across all of the variables. Figure 1 (adapted from Weiss [1]) illustrates the cost model for product lines versus one-at-a-time systems. Without product lines, the cumulative cost begins at zero and rises steadily. With product lines, the cumulative cost begins with the initial cost of the core assets, and then climbs at a shallower rate as projects are turned out. The point at which the two lines cross (and the one-at-a-time cost curve becomes higher) is the payoff point. That cross-over point can be expressed in terms of the number of systems built. Thus, the question becomes how many systems do we need to field before the savings from reuse pay for the up-front investment in building the core asset base? Surprisingly, the answer to this question does not seem to vary much across organizations, products, or markets. What do you imagine it is? Five or six, maybe? Ten? Well, it turns out for the product line approach to pay off, experts agree that the number of systems you need to build is…

Two. Maybe three.

That's all. And that's a very good answer indeed, because it's hard to imagine a product line without at least two or three family members. This means that, practically speaking, every single product line can be expected to reap cost savings when compared to building and maintaining its constituent products separately. Who says so? Apparently, just about everyone:

- In their book *Software Product Line Engineering*, David Weiss and Robert Lai write that the FAST process of product line engineering produces a payoff after about three systems [1]. Subsequent to the book's publication, Weiss reports new data from Lucent Technologies software product lines that suggests the payoff point is usually between two and three. Audris Mockus and Harvey Siy confirm this with excellent data that shows that for a FAST-based product line at Lucent called AIM, "the first nine months of applying AIM saved around three times more effort (61/21) than was spent on implementing AIM itself" [2]. (Savings were $6-9,000,000.)
- Jacobson et al. report in *Software Reuse: Architecture, Process, and Organization for Business Success* [3] that a component used three to five times in application projects will recover the initial cost of creating it and the ongoing cost of supporting it. It costs 1.5 to 3.0 times as much to create and support a reusable component as it does to implement a similar component for a single application (suggesting that the payoff point is 1.5 to 3.0 projects). It costs only one quarter as much to utilize a reusable component as it does to develop a new one from scratch. It takes two or three product cycles, they say, before the benefits of reuse become significant.
- In *Measuring Software Reuse*, Jeff Poulin writes that large-scale strategic reuse (the essence of a product family) becomes worthwhile after the assets are used in a total of three applications [4]. Other reuse experts seem to agree. Don Reifer says the cost recovery point is three in *Practical Software Reuse* [5], as does Will Tracz in *Confessions of a Used Program Salesman* [6], and Ted Biggerstaff (reported by Tracz in a reuse workshop summary [7]). The data of Jacobson et al., cited above, are completely consistent with this estimate.
- John Gaffney and Bob Cruikshank of the Software Productivity Consortium published data in 1992 about SPC's synthesis method of domain and application engineering. Their data, ranging across three application domains, showed that the payoff point for product-line-style reuse was between 1.67 and 4.86 systems [8]. They also showed that return-on-investment is based on the payoff point: ROI = (N/N0 – 1)100%, where N is the number of systems built and N0 is the payoff point.
- A major American aerospace company has told us they believe the payoff point for their product line of avionics systems comes after about two major block upgrades. A block upgrade is a new, substantially-revised version of one of the avionics systems in their family – in essence, a new family member.

The reuse figures would not seem to take into account the overhead of organizational change that is inherent in adopting a product line approach, and so might seem a little low – at least at first glance. But organizations that are aggressive and disciplined about reuse (even if they don't call what they're doing a product line) must have worked out responsibilities for building the reusable assets and insuring their reuse in a disciplined way, and so their payoff point reflects the

organizational re-structuring necessary to achieve that. So accepting their payoff point at face value will probably not take us too far off the mark. And to reinforce that, the Lucent, SPC, and aerospace examples do take the organizational costs into account, and their payoff points are even a bit lower.

This result is very encouraging. It's almost a certainty that if you're planning a product line, you're planning to have at least three products in it. So a slightly different flavor of our opening question is "Will I save money by adopting a product line approach?" And the answer is a very reassuring "Almost certainly."

Now we aren't promising that you'll *make* money. People have to actually buy your products for that to happen, and even the most breathtakingly efficient production process can't guarantee that. Instead, we're talking about the relative costs of building a set of products as a product line, versus building the same set as unrelated one-of-a-kind products. The point here is that if you're planning to build three or more related systems – not a particularly burdensome requirement for a product line -- then building them from a common set of core assets is almost guaranteed to be the more economical way to turn them out.

References

1. Weiss, D. M. & Lai, C. T. R. Software Product-Line Engineering: A Family-Based Software Development Process. Reading, MA: Addison-Wesley, 1999.

2. Mockus, A. & Siy, H. "Measuring Domain Engineering Effects on Software Coding Cost," 304-311. Proceedings of Metrics 99: Sixth International Symposium on Software Metrics, Boca Raton, FL, November, 1999. New York, NY: IEEE Computer Society Press, 1999.

3. Jacobson, I.; Griss, M.; & Jonsson, P. Software Reuse: Architecture, Process, and Organization for Business Success. New York, NY: Addison-Wesley, 1997.

4. Poulin, J. S. Measuring Software Reuse: Principles, Practices, and Economic Models. Reading, MA: Addison-Wesley, 1997.

5. Reifer, D. J. Practical Software Reuse: Strategies for Introducing Reuse Concepts in Your Organization. New York, NY: John Wiley and Sons, Inc., 1997.

6. Tracz, W. Confessions of a Used Program Salesman: Institutionalizing Software Reuse. New York, NY: Addison-Wesley, 1995.

7. Tracz, W. "RMISE Workshop on Software Reuse Meeting Summary," 41-53. Software Reuse: Emerging Technology. Los Alamitos, CA: IEEE Computer Society Press, 1998.

8. Gaffney, J. E. & Cruickshank, R. D. “A General Economics Model of Software Reuse,” 327-337. Proceedings of the 14th ICSE, Melbourne, Australia, May 11-15, 1992. New York, NY: ACM, 1992.

About the Author

Dr. Paul Clements is a senior member of the technical staff at Carnegie Mellon University's Software Engineering Institute, where he has worked for 8 years leading or co-leading projects in software product line engineering and software architecture documentation and analysis.

Clements is the co-author of three practitioner-oriented books about software architecture: *Software Architecture in Practice* (1998, second edition due in late 2002), *Evaluating Software Architectures: Methods and Case Studies* (2001), and *Documenting Software Architectures: View and Beyond* (2002). He also co-wrote *Software Product Lines: Practices and Patterns* (2001), and was co-author and editor of *Constructing Superior Software* (1999).In addition, Clements has also authored dozens of papers in software engineering reflecting his long-standing interest in the design and specification of challenging software systems.

He received a B.S. in mathematical sciences in 1977 and an M.S. in computer science in 1980, both from the University of North Carolina at Chapel Hill. He received a Ph.D. in computer sciences from the University of Texas at Austin in 1994.

Watts New?

Some Programming Principles: People

WATTS S. HUMPHREY

This is the fourth and last in a series of columns about programming principles. The prior columns discussed the principles that relate to programming requirements, software products, and the projects for developing these products. This column deals with people and the human aspects of the software process. While this is an enormous subject and no brief column could possibly do justice to the vast body of relevant knowledge, a few key principles are particularly important in determining the performance of software people and the teams on which they work.

While most people behave in reasonably predictable ways, software people are unique, both in their creative abilities and in the nature of the work they do. Software professionals are among the brightest people on earth. Most of us got into this field because we were excited by the thrill of working with a challenging and very special technology. However, the problem many of us face is that the environment in which we work rarely supports and motivates consistently high-quality creative work.

In addressing this subject, I discuss the factors that govern the performance of software professionals, the most effective ways to obtain superior performance, and the key issues to consider in motivating and guiding teams of creative professionals.

The Performance of Software Professionals

Much as in other professions, the performance of software people is governed by four things.

- their understanding of the job they have to do
- their knowledge of and skill at using the best known methods for that job
- their discipline to properly and consistently use their knowledge and skill
- the quality of the support system that motivates and controls their activities

These four governing factors form the basis of the four people principles discussed in this column.

People Principle Number 1: If the programmers do not understand the job they are to do, they will not do it very well.

This seems to be such an obvious point that it is hardly worth discussing. However, it is critically important and often overlooked. If the members of a development team are not intimately familiar with the job their product is to perform and the way the users of that product will use it, the project will almost certainly be troubled and the product will likely be a failure. If you can't put users or people with user experience on your development team, at least ensure that the team has ready

access to people with such knowledge. To produce quality products, a close and cooperative relationship with such people is absolutely essential.

People Principle Number 2: The people who know and use the best methods will do to best work.

While software developers usually get extensive training on tools and methods, they generally get little or no guidance on their personal practices. This is not true of any other sophisticated technical field. Chemical engineers are not born knowing how to conduct experiments, analyze the composition of materials, or follow sound laboratory practices. Doctors learn their profession through extensive training, by completing several years of internships and specialty studies, and by mastering the methods that their predecessors have found most effective.

In software, we have yet to learn the truisms that some methods and practices are more efficient and cost-effective than others and that the cost and quality of the products we produce is governed by the practices we use in developing them. Today, on many projects, the developers do very similar work but their personal practices are very different. I have studied such teams and found that even developers who do similar work use different methods and, what is worse, they are generally unaware of the methods their peers are using. Except for occasional help with problems or complex tools, most software people work largely alone and are unaware of how others work or the best ways to do each of their tasks.

If every scientist had to personally discover Bernoulli's principle, develop Maxwell's equations, or invent calculus, we would still be in the dark ages. The explosive growth of science and engineering didn't start until people defined their practices, measured their work, and communicated precisely. This allowed others to repeatably produce the same results. The essence of science and engineering is learning from the experiences of others. Until we build a body of professional software practices and teach new professionals to consistently and properly use these practices, programming will remain an unsatisfactory craft that produces defective, insecure, and even unsafe products.

People Principle Number 3: When programmers know how to select and consistently use the best methods, they can do extraordinary work.

We now have data on several thousand programmers who have taken the Personal Software Process (PSP) course as well as data on over 100 Team Software Process (TSP) teams that have been launched[1]. It is now clear that developers can learn and use highly-effective personal practices and that, when they use these practices, they produce much better work than they ever did before. In a recent study of 20 TSP teams that provided data on delivered product defect levels,

1. TSP team projects start with a launch where the members learn the project requirements, define their own processes, and develop and negotiate their plans with management.

these products had 100 times fewer defects than average industrial products and 16 times fewer defects than typical products produced by CMM level 5 organizations [1]. In most cases, these were first-time TSP teams. Since personal and team performance typically improves with experience, we can expect even better performance in the future.

The obvious problem with requiring that developers use the best practices is in determining what the best practices and methods are. However, the reason that the PSP and TSP are so effective is that they provide developers the tools and methods they need to make this decision for themselves. With the PSP, professionals learn how to follow a defined process, how to modify that process when they need to, and how to measure and plan their personal work. By using their own data, developers can see what methods work best for them and they can make informed decisions about how to do their work.

Similarly, when TSP teams are launched, they examine the job that they have to do and consciously decide on the best strategy, process, and plan for the work. While this may not seem to be the best possible way to do the job, it is the one that the team members think would be best, and these are the only people who know their personal skills, abilities, and interests; the work that must be done; and how they can best work together as a team. So, while there may be theoretically better ways to do the work, the team's informed decision on its own strategy, process, and plan will actually produce the best way for this team to do this job.

People Principle Number 4: Superior software work is done by highly motivated developers.

When people are discouraged, antagonized, or even just unhappy, they cannot do their best work. The key to getting superior work from creative people is to energize the entire team and to motivate all of the members to do their very best. But what motivates software people and how can one build and sustain this motivation? In an interesting study of software projects, Linberg compared management's typical views of project success with those of the team members [2]. While managers typically think in terms of cost, schedule, and product success, the developers viewed their projects quite differently.

For example, Linberg asked one group of experienced developers what project they viewed as the most successful one on which they had worked. They had just completed what he referred to as project A and over half of them cited this as the most successful of their careers. This was in spite of the fact that they had all worked on 8 or more projects and that this job was delivered in twice the desired time and for three times the planned cost. The four factors that the team members listed as making this project successful were as follows.

- a personal sense of being involved and making a contribution
- frequent celebrations where the team and management complemented them on their achievements and milestones

- positive feedback from marketing and senior management
- the autonomy to do the job the way that they thought was best

These are the things that motivate software developers. While these same factors motivate people in almost all walks of life, they are particularly important for getting superior software work. In many fields, people's personal practices are visible and relatively easy to measure and monitor. In software, much of the work is intellectual and not measurable or manageable without the developer's cooperation. This is why motivation, personal discipline, and sound professional behavior is critically important for software development. If software people do not want to work in a particular prescribed way, they won't and, unless the software people themselves tell them, it is unlikely that anyone will know. This is why contributing, being involved, being rewarded, getting positive feedback, and having autonomy are particularly important for software developers.

Whether you manage software development or do development work yourself, remember and follow these four principles. To produce superior products, the developers must

- understand the job the product is to do
- know the best methods for doing the work
- consistently select and use these best methods
- be highly motivated to work on this team doing this job

Acknowledgements

In writing papers and columns, I make a practice of asking associates to review early drafts. For this column, I particularly appreciate the helpful comments and suggestions of Dan Burton, Anita Carleton, Julia Mullaney, Bill Peterson, and Marsha Pomeroy-Huff.

In closing, an invitation to readers

In these columns, I discuss software issues and the impact of quality and process on developers and their organizations. However, I am most interested in addressing the issues that you feel are important. So, please drop me a note with your comments, questions, or suggestions. I will read your notes and consider them when planning future columns.

Thanks for your attention and please stay tuned in.

Watts S. Humphrey
watts@sei.cmu.edu

References

[1] Davis, N. & Mullaney, J. *The Team Software Process (TSP) in Practice: A Summary of Recent Results* (CMU/SEI-2003-TR-014). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. <http://www.sei.cmu.edu/publications/documents/03.reports/03tr014.html>.

[2] Linberg, K. "Software Developer Perceptions about Software Project Failure: A Case Study." *Journal of Systems and Software 49,* 2 (December 1999): 177-192.

About the Author

Watts S. Humphrey founded the Software Process Program at the SEI. He is a fellow of the institute and is a research scientist on its staff. From 1959 to 1986, he was associated with IBM Corporation, where he was director of programming quality and process. His publications include many technical papers and six books. His most recent books are *Managing the Software Process* (1989), *A Discipline for Software Engineering* (1995), *Managing Technical People* (1996), and *Introduction to the Personal Software Process*^{SM} (1997). He holds five U.S. patents. He is a member of the Association for Computing Machinery, a fellow of the Institute for Electrical and Electronics Engineers, and a past member of the Malcolm Baldrige National Quality Award Board of Examiners. He holds a BS in physics from the University of Chicago, an MS in physics from the Illinois Institute of Technology, and an MBA from the University of Chicago.

Features

CMMI Adoption Trends

LAUREN HEINZ

Since the release of the CMMI® Product Suite in January 2002, organizations in the systems and software communities have demanded credible evidence about the adoption, impact, and benefits of CMMI-based process improvement.

Until recently, little of such evidence was publicly available. In November, however, the SEI published the first CMMI Maturity Profile, which shows the latest CMMI adoption trends, and a special report describing how several organizations are implementing CMMI models with significant positive results. Both publications are available through the SEI℠ Web site.

Initial appraisal data reported to the SEI show that a variety of organizations from different industries and environments are rapidly upgrading to the new models, that users are implementing the full range of model scopes and representations offered, and that organizations around the globe are enjoying high levels of process maturity through use of CMMI models.

Rapid Adoption Worldwide

A maturity profile shows how an improvement method, such as a CMMI model, is being adopted worldwide based on SEI-authorized appraisal results. Data are presented in a series of graphs and bar charts by organization type, size, location, maturity level, and other characteristics. The first CMMI Maturity Profile is based on 100 Standard CMMI Appraisal Method for Process Improvement (SCAMPI℠) V1.1 appraisals using CMMI V1.1 models that were conducted between April 2002 and June 2003.

According to SEI data, 93 different organizations have performed a CMMI appraisal (some more than once). A breakdown by "organizational type" follows:

- Commercial/In-House
  47.3%
- DoD/Federal Contractor
  45%
- Military/Federal
  7.7%

"The above figures show roughly equal adoption of CMMI by organizations in the commercial and government sectors," says Dave Zubrow, the SEI manager who helps to produce maturity profiles. Regarding the size of the organizations conducting these appraisals, 50% have from 1 to

200 employees, and 50% have 201 or more employees—reflecting use among small and medium-to-large organizations alike.

Another view of adoption is to look at the geographic distribution. The countries and the number of organizations appraised using CMMI models follows:

Country	Number
US	46
Japan	25
India	10
United Kingdom	5
Australia	3
China	2
France	3
Taiwan	2
Canada	1
Russia	1
South Korea	1
Switzerland	1

These data show that 54, or 54%, of the appraisals were conducted from outside of the United States. As a point of comparison, 52% of the organizations included in the latest Capability Maturity Model for Software (SW-CMM) Maturity Profile are from outside of the United States. "Apparently, CMMI is achieving a level of international adoption in the short time that it has been available comparable to that recently reached by the SW-CMM," Zubrow said. The SW-CMM has been available for about 10 years.

A Variety of Scopes and Representations

Organizations can choose from several different CMMI models when designing a process improvement program, including CMMI for Software, CMMI for Systems and Software Engineering (SE/SW), CMMI SE/SW with integrated product and process development (IPPD), and CMMI SE/SW with IPPD and Supplier Sourcing. Zubrow said that every variety of model was represented in the maturity profile.

There are also two model representations from which organizations may choose: staged and continuous. The staged approach predefines the process areas required to attain each maturity level (1-5) and thereby provides a roadmap for institutionalizing best practices. Organizations that are upgrading from the SW-CMM, a staged-only model, are likely to prefer staged.

In the continuous representation, process areas are organized into four process area categories: Process Management, Project Management, Engineering, and Support. Based on its business objectives, an organization selects the process areas in which it wants to improve and to what degree. Instead of organizational maturity levels, capability levels (0-5) for each process area are used to measure improvement.

Since the release of CMMI V1.1, approximately one-third of the Introduction to CMMI course attendees have selected the continuous version of the course, and nearly one-fourth of all SCAMPI appraisal results reported to the SEI have been from organizations using the continuous representation.

Other Findings

Zubrow says that, overall, the new CMMI Maturity Profile depicts a relatively mature set of organizations. "The data suggest that the early adopters tend to be a more process-mature group than the community at large," he says. "The CMMI Product Suite with its options and flexibility should reach a broader audience and help create a global community of process improvement for those involved in the development, maintenance, and acquisition of software-intensive systems."

The CMMI Maturity Profile is available for download at *http://www.sei.cmu.edu/sema/profile.html?si*.

Benefits Case Studies

The special report that was recently published, *Demonstrating the Impact and Benefits of CMMI: An Update and Preliminary Results*, describes the experiences of organizations that have decided to implement CMMI. The 12 case studies in the report, covering organizations such as Accenture, the Boeing Company, General Motors, and Bosch, demonstrate the impact that CMMI-based process improvement has on each organization's performance. The case studies feature initial evidence that adoption of CMMI can result in decreased project costs, increased schedule predictability, improved product quality, increased customer satisfaction, and a positive return on investment.

The report is available at *http://www.sei.cmu.edu/publications/documents/03.reports/03sr009.html?si*.

For more information, contact—

Customer Relations

Phone
412-268-5800

Email
customer-relations@sei.cmu.edu

World Wide Web
http://www.sei.cmu.edu/cmmi/?si

CERT's Function Extraction Project: Exploring Program Behavior for Security Analysis

JANET REX

Substantial computer programs are often remarkably complex in their structure and behavior. When faced with this complexity, software engineers have no practical means to quickly determine the full functional behavior of programs in all circumstances of use. The result is that today we must depend on large-scale systems to function correctly, even though they are composed of programs whose full behavior and security exposures may not be reliably known. The existence of unknown functionality—and the malicious exploitation of that functionality—is the Achilles heel of software. The ability to automatically compute the behavior of software quickly and comprehensively would help to solve the problem.

While computing software behavior is a very difficult task that poses many theoretical and engineering challenges, the benefits could be substantial. For that reason, the SEI's CERT® Research team has begun early, exploratory investigations to discover the extent to which software behavior can be calculated. Of particular interest is the possibility of using behavior calculation for malicious code detection and analysis.

Understanding Program Behavior

Understanding program behavior today is a haphazard, error-prone, resource-intensive process. Nevertheless, it is essential for uncovering security and reliability problems. And because attackers can make malicious modifications to programs at any time, the task of behavior discovery never ends. Many programs are difficult to understand because they contain an enormous number of execution paths, any of which might contain errors or security exposures. Faced with massive sets of possible executions, programmers can often do no more than achieve a general understanding of mainline program behavior.

Automated support for behavior discovery would be an ideal solution to this difficult problem. Members of CERT Research are exploring this strategy in an effort called the function extraction (FX) project. The objective of the project is to investigate technology to help move from an uncertain understanding of program behavior derived slowly by humans to a precise understanding quickly calculated by computers.

Treating Programs Like Equations

Traditional engineering disciplines depend on rigorous methods to evaluate the expressions that represent and manipulate their subject matter. For example, the equations that represent fluid mechanics, or electromagnetic fields, allow engineers to understand and predict the behavior of their designs with precision. As an engineering discipline, software engineering has been an

exception to this rule; it traditionally has had no way to quickly and completely evaluate the expressions it produces. In this case, the "expressions" are computer programs, and "evaluation" means understanding their full behavior in every instance—behavior that is right or wrong, behavior that is the intention of the engineer or the result of malicious intervention.

The function-theoretic model of software treats programs as implementations of mathematical functions or relations, that is, mappings from inputs (stimuli) to outputs (responses), no matter what subject matter they deal with. The key to the function-theoretic approach is the recognition that although the sequential logic of programs can contain an immense number of execution paths, this logic is also composed of a finite number of control structures, each of which implements a mathematical function or relation in the transformation of its inputs into outputs.

This finite property of program logic, viewed through the lens of function theory, suggests the possibility of automated calculation of program behavior. Every control structure in a program has a non-procedural behavior signature that defines its net functional effect. Behavior signatures can be extracted and composed with others in a stepwise process. The resulting behavior signature of an entire program represents the specifications or business rules that it implements. Initial investigation of these concepts suggests that theoretical challenges to function extraction may have acceptable engineering solutions. For example, while no general theory for loop behavior calculation can exist, pattern recognition can help provide an engineering approach.

Moving Forward

The CERT Research team plans to investigate function extraction methods and to create a demonstration prototype. Foundations for function extraction are being developed, and a pre-prototype function extractor has been created that calculates the behavior of programs written in a small subset of Java. Participation by organizations in government, defense, and industry is welcomed, by partnering with CERT Research as a visiting scientist, or through our Affiliate Program.

Exploration at the SEI

The function extraction (FX) project is an excellent example of the exploratory research that is done at the SEI. In the life-cycle model used by the SEI to describe new technologies to our collaborators, the FX project falls into the earliest phase: exploration. In the exploration phase, the SEI identifies potentially high-payoff approaches to DoD needs and other pervasive problems in the software community at large. While the SEI provides leadership during the exploration phase, we also try to identify partners who can work with us to develop a solution. (See Eileen Forrester's article[1] in the 3rd quarter issue of news@sei interactive for more information on the SEI's life-cycle approach to technology transition.)

Some practitioners are well suited to joining the SEI in ventures like the FX project, because they enjoy shaping the problem space and acting as co-developers. If you identify yourself as an innovator for this technology, consider joining forces with us on this (or other) exploratory efforts.

For more information on collaborating with the SEI, contact SEI Customer Relations at 412-268-5800 or *customer-relations@sei.cmu.edu*.

[1] http://interactive.sei.cmu.edu/news@sei/features/2003/3q03/feature-4-3q03.htm

For more information, contact—

Richard C. Linger,
Principal Investigator

Phone
412-268-5800

Email
rlinger@sei.cmu.edu

World Wide Web
http://www.sei.cmu.edu/?ns

Strategic Technology Transition: A New Kind of Partnership

JANET REX

The SEI and the U. S. Army Aviation and Missile Command (AMCOM) Software Engineering Directorate (SED) have entered into a partnership that establishes AMCOM as the leader in technology transition for SEI technologies for the Army. This special partnership was marked by the opening of an SEI satellite office, the first of its kind, in Huntsville, AL, in the fall of 2002.

To help connect SEI technologies with AMCOM's future business, the SEI conducted two strategic business and technology planning workshops with SED leadership. These workshops drew many top-level SED managers, including William Craig, director of the SED, and several of his direct reports, along with key program managers and technical staff. The workshops resulted in a mapping of relevant SEI technologies to SED business objectives.

The AMCOM SED has already reported achievement of maturity level 4 in the Capability Maturity Model® for Software. As one of the early partnering projects, the SEI is supporting AMCOM in moving to Capability Maturity Model Integration (CMMI®), as well as in adopting use of the Personal Software Process℠ (PSP℠) and the Team Software Process℠ (TSP℠) in software engineering projects. Once these practices are in place at the SED, they will provide a foundation for the transition of relevant SEI technologies to other Army organizations worldwide. Other SEI software engineering management and technology practices, along with SEI product line and COTS acquisition practices, will also be considered for transition to the SED.

Since the AMCOM SED provides critical support to AMCOM, the Army Materiel Command, the National Aeronautics and Space Administration (NASA), and the Army acquisition and software engineering communities, the SED's commitment to improving its software engineering and acquisition practices benefits many constituencies. Scott Reed, manager of the SEI field office in Huntsville says, "Civil agencies (such as NASA) will also have the opportunity to collaborate with the SED for adoption of practices that will allow them to improve their technical and management capabilities."

Strategy for Widespread Improvement

AMCOM SED also plans to transition technology to colleges and universities in the Huntsville area. They will then have the opportunity to build engineering curricula to better support the local needs of AMCOM and industry. Also, regional businesses will have the opportunity to collaborate with the SED in the adoption of technologies and practices that will best benefit them.

For example, a joint pilot project has been run to determine the feasibility of developing guidance and other aids to support adoption of CMMI by small and medium enterprises (SMEs). For this pilot, the SMEs were required to be companies with 25 to 250 employees located in the Huntsville

area. The two companies chosen to participate are involved in product development for government or in engineering services to support product development. Both had recent experience with ISO 9000. The pilot implemented three CMMI process areas at these companies to help codify recommendations for how to package, sell, appraise, and implement CMMI for SMEs.

The pilot project has provided the SEI with some data on articulating the business case for small companies to adopt CMMI, and has also supplied SEI CMMI projects with feedback from the field. A workshop titled "What Have We Learned from Huntsville Pilots?" is scheduled for March 29 at the Southeastern Software Engineering Conference to share artifacts, processes, and results from the pilots. For more information, follow the CMMI Workshop link at *http://www.ndia-tvc.org/SESEC*.

The Benefits of Partnership

Some of the anticipated benefits of this partnership include

- improved engineering practices throughout the system life cycle, including research and development
- improved systems acquisition practices for the Army Program Offices
- improved success rates for inserting technology into Army programs
- creation of curricula in academia that support Army engineering needs
- strengthening of regional defense contractors' engineering capabilities for better-quality products

Using transition techniques and products developed by the SEI, the SED will structure effective transition and adoption programs for Army organizations. This will help to make technology transition more predictable, a consistent stream of continuous improvement, providing benefits to a wide range of both users and developers, both regionally and within the Army.

SEI Strategic Impact Programs

Since its inception, the SEI has been helping U.S. government acquisition programs in their efforts to improve their processes and minimize risks. Recently, the SEI formalized this ongoing support by creating the Acquisition Support Program[1] (ASP), a group devoted to addressing the unique demands and challenges of acquisition. In Fiscal Year 2002 the SEI also established strategic impact programs (SIPs) for each military service (Air Force, Army, and Navy). A SIP is a multi-year program of work, a strategic commitment to improvement and change within a particular acquisition community and industry base. The goal is to contribute to the success of acquisition programs that fall within the scope of a military service SIP. Delivery teams focus on understanding and meeting the needs of programs within a SIP. The field office in Huntsville is an important part of the Army SIP.

The Air Force SIP is working closely with the Space and Missile Center and the Electronic Systems Center. The Department of the Navy SIP is working with Navy and Marine Corps organizations, identifying support and transition opportunities. While neither of these SIPs has established a field office like the one in Huntsville, both are developing relationships to demonstrate the applicability and relevance of SEI technology to support the development and acquisition of software-intensive systems in the military.

[1] http://www.sei.cmu.edu/programs/acquisition-support/index.html?si

For more information, contact—

Scott Reed

Phone
412-268-5800

Email
lsr@sei.cmu.edu

World Wide Web
http://www.sei.cmu.edu/about/overview/directions/huntsville/?si

New Credentials Program Developed by SEI Education and Training

ERIN HARPER

Since 1984 the SEI has been identifying, developing, and advocating practices for designing high-quality software and protecting networked systems. To help organizations put these practices in place, the SEI has developed a new Credentials Program. The certificates and certifications developed for this program guide participants through a series of courses chosen to help them develop expertise in a specific area of work.

Certificates are awarded after participants attend a specified series of courses and serve to recognize the successful completion of an educational process. Certificate programs are a good way to build skills and generally do not require testing or follow-up training. Certification is earned after the completion of a specified series of courses and an assessment against a set of industry-relevant standards. Certification often includes ongoing requirements that must be met to keep the certification valid. While SEI certifications do not grant permission to use the intellectual property of the SEI, they do signify that the student has been certified by the SEI to have obtained a specific set of skills and knowledge in a particular area. Certification allows participants to build their credentials through an objective confirmation of their skills.

Currently, the SEI offers seven certificate programs and one certification program, although other programs are in development. “My job at the SEI is to help the technical programs define a logical path through the SEI curriculum for participants,” says Michael Carriger, the team lead for the SEI’s educational programs. “While the SEI’s course offerings have been well received for many years, these programs add value to our courses by showing which work best together.”

SEPM Certificate Programs

Four Software Engineering Process Management (SEPM) certificate programs are designed for those involved in managing, leading, or participating in process improvement efforts.

The SEI Certificate in Software Engineering Process Management is for managers who need a toolbox of approaches to managing and improving the software engineering process, providing the knowledge and skills necessary for a solid foundation in process improvement. The program explores management, metrics, and Personal Software Process^{SM} (PSP^{SM}) and Team Software Process^{SM} (TSP^{SM}) approaches to process improvement.

For those leading change efforts within their organizations, the SEI Certificate in Software Process Improvement Implementation provides a look at several different methods for implementing change, including the IDEAL model (a high-level description of the phases of process improvement), metrics, and TSP/PSP. This program provides change agents (such as software engineering process group [SEPG^{SM}] members and leads) with the information they need if their

organizations are planning to implement quality initiatives, such as Capability Maturity Model Integration (CMMI®).

Using CMMI, organizations can improve their ability to develop and maintain quality products and services. The SEI Certificate in CMMI guides change agents and managers through the fundamental concepts of CMMI and the relationships among CMMI model components; the IDEAL model; and the use of statistical process control to manage and improve software processes.

While CMMI provides a powerful improvement framework that helps organizations understand what needs to be done to improve processes, it doesn't specify how this should be done. The PSP provides a roadmap for organizations and individuals to follow, and the SEI Certificate in Personal Software Process for CMMI helps engineers learn how to implement CMMI using the PSP.

Computer Security Incident Handler Certification Program

Organizations need individuals who can lead computer security incident response teams. The demand is accelerating as technology advances, applications gain complexity, services become more regulated—and as new regulations mandate increasingly disciplined response capabilities. In response to this demand, the SEI CERT® Coordination Center (CERT/CC) has introduced a program designed to train and certify computer security incident handlers. Participants must have three years of technical or managerial experience in incident handling and submit an application and a letter of recommendation. In addition to completing the required courses, participants must pass an evaluation exam to become certified.

Software Architecture Certificate Programs

Based on decades of experience with software-intensive systems and supported by four widely acclaimed practitioner books in the SEI Addison-Wesley Series[1], the SEI has developed three software architecture certificate programs to equip software professionals with state-of-the-art practices for designing, documenting, evaluating, and implementing software architectures.

Beginning with an introduction to software architecture fundamentals, the Software Architecture Professional Certificate helps practitioners gain experience in architecture documentation, design, and analysis techniques. The program also shows how these techniques can be used effectively with a product line approach, in which a set of software-intensive systems are developed from a common set of core assets in a prescribed way.

The Architecture Tradeoff Analysis Method℠ (ATAM℠) is a method developed by the SEI for evaluating software architectures relative to the quality attributes that are most desirable in a

1. http://www.sei.cmu.edu/publications/books/sei.series.html?si

particular system. Qualified participants who complete the ATAM Evaluator Certificate are authorized by the SEI to participate in architecture evaluations using the ATAM.

The ATAM Lead Evaluator Certificate program provides qualified participants with the technical depth, social techniques, and experience they need to effectively lead software architecture evaluations using the ATAM. In addition to completing the required courses, participants must successfully lead an ATAM architecture evaluation observed by an SEI ATAM expert. SEI-authorized lead evaluators must attend yearly ATAM update workshops to maintain their skills and status.

Completing an SEI certification or certificate program gives software engineers and process improvement professionals official recognition of their skills and expertise and costs less than enrolling in the courses individually. For program prices or to get started in a program, please contact the SEI.

For more information, contact—

Michael Carriger

Phone
(412) 268-4469

Email
msc@sei.cmu.edu

World Wide Web
http://www.sei.cmu.edu/products/courses/certificates/?si