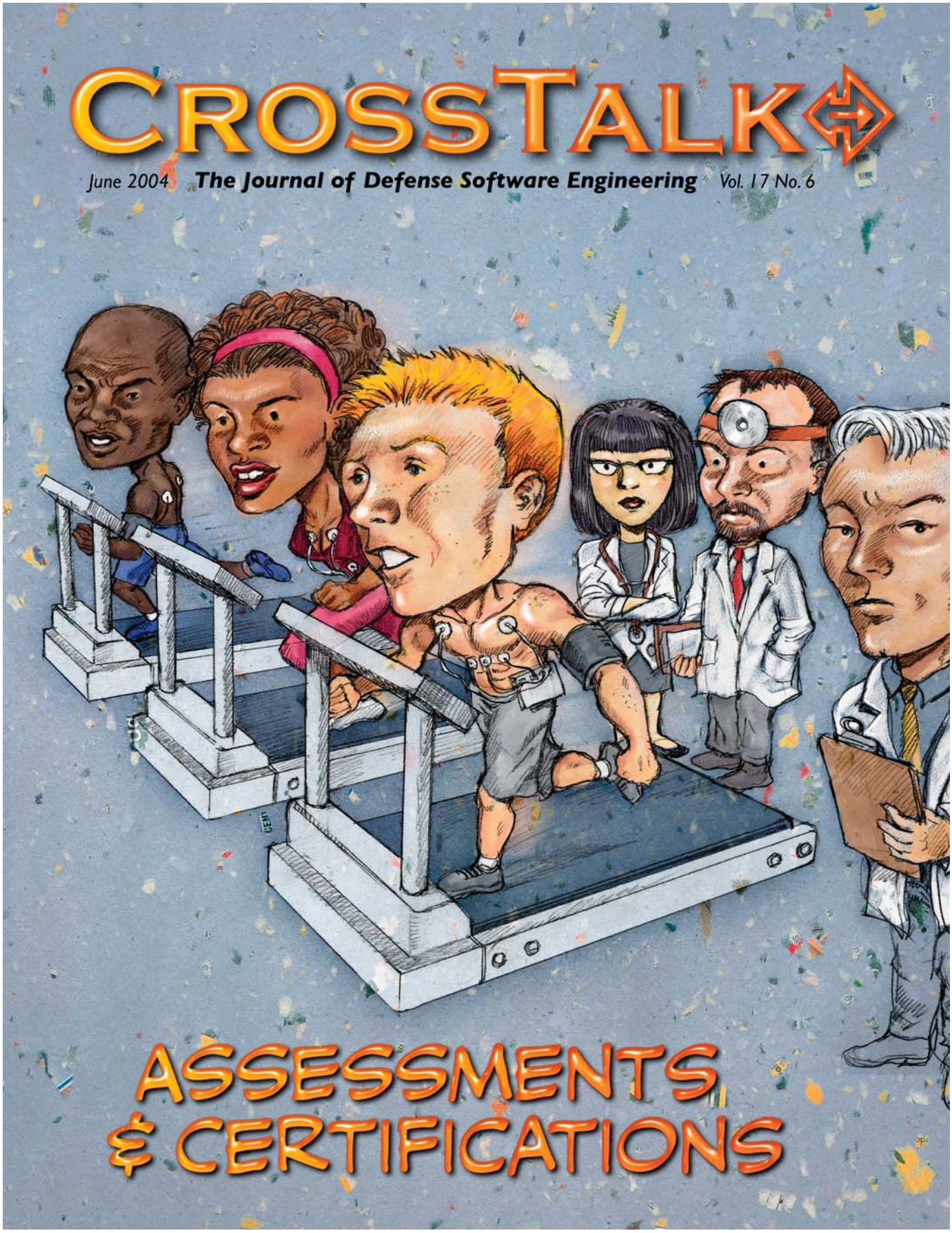
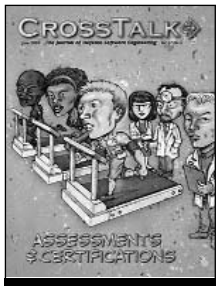
CROSSTALK

June 2004 The Journal of Defense Software Engineering Vol. 17 No. 6



ASSESSMENTS & CERTIFICATIONS

Assessments and Certifications



ON THE COVER
Cover Design by Kent Bingham.

Software Engineering Technology

Open Forum

Online Article

Departments

CROSSTALK

PUBLISHER	*Tracy Stauder*
ASSOCIATE PUBLISHER	*Elizabeth Starrett*
MANAGING EDITOR	*Pamela Palmer*
ASSOCIATE EDITOR	*Chelene Fortier-Lozancich*
ARTICLE COORDINATOR	*Nicole Kentta*
CREATIVE SERVICES COORDINATOR	*Janna Kay Jensen*
PHONE	(801) 586-0095
FAX	(801) 777-8069
E-MAIL	crosstalk.staff@hill.af.mil
CROSSTALK ONLINE	www.stsc.hill.af.mil/crosstalk

Subscriptions: Send correspondence concerning subscriptions and changes of address to the following address. You may e-mail or use the form on p. 29.

Ogden ALC/MASE
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 and Permissions: Requests for reprints must be requested from the author or the copyright holder. Please coordinate your request with CROSSTALK.

Trademarks and Endorsements: This DoD journal is an authorized publication for members of the Department of Defense. Contents of CROSSTALK are not necessarily the official views of, or endorsed by, the government, the Department of Defense, or the Software Technology Support Center. All product names referenced in this issue are trademarks of their companies.

Coming Events: We often list conferences, seminars, symposiums, etc. that are of interest to our readers. There is no fee for this service, but we must receive the information at least 90 days before registration. Send an announcement to the CROSSTALK Editorial Department.

STSC Online Services: www.stsc.hill.af.mil
Call (801) 777-7026, e-mail: stsc.webmaster@hill.af.mil

Back Issues Available: The STSC sometimes has extra copies of back issues of CROSSTALK available free of charge.

The Software Technology Support Center was established at Ogden Air Logistics Center (AFMC) by Headquarters U.S. Air Force to help Air Force software organizations identify, evaluate, and adopt technologies to improve the quality of their software products, efficiency in producing them, and their ability to accurately predict the cost and schedule of their delivery.

Know Where Your Organization Stands Today and Determine How to Improve for Tomorrow

At the Air Force Software Technology Support Center (STSC), CrossTalk's parent organization, some of our most requested services are our assessment services. When describing these assessment services, we often use a comparison of going to the doctor for an annual check-up. An STSC assessor or a team of assessors diagnoses the health of an organization and provides an overall rating. STSC assessors present their findings and provide suggestions on improvement but it is up to the organization to act on implementing the necessary process improvement practices to better the health of their organization. The *doctor* (assessor) can't do that for them.

That is the focus of this month's issue: Assessments and Certifications. A variety of doctors and diagnostic techniques exist in the systems and software field today. Whether the doctor specializes in ISO 9001, Capability Maturity Model® (CMM®), Capability Maturity Model Integration℠ (CMMI®), Federal Aviation Administration integrated Capability Maturity Model (FAA-iCMM®), or is more of a general practitioner, there are many standards and models to help him or her assess an organization. These standards and models become the core or foundation by which a company's project and organizational processes are built. So together, these standards and models are critical in our understanding, whether assessors, acquirers, managers or practitioners, of where an organization stands today to determine how to improve for tomorrow.

With over 500,000 companies worldwide registered to ISO 9001, we begin this month's issue with a look at this quality management system standard. In *Why Be Assessed to the Most Prevalent Standard in Use Today?*, Robert Vickroy provides helpful information to companies just beginning their ISO 9001 continuous process improvement journey.

Next, in *CMMI Myths and Realities*, Lauren Heinz discusses how the CMMI Product Suite has helped hundreds of organizations improve their processes despite initial concerns regarding the size and complexity of the model as well as what type of organization the CMMI might be best suited for.

The Federal Aviation Administration (FAA) and Lockheed Martin have both been developing and implementing integrated process improvement frameworks, specifically the FAA-iCMM and the Integrated Engineering Process standard, respectively. In *There Is More to Process Improvement Than Just CMM*, Dr. Linda Ibrahim and Joan Weszka explain how integrating existing models and standards can address broad, enterprise-level process improvement. Appraisal methods for these integrated models are also discussed.

Don't miss this month's supporting articles, *Predictable Assembly From Certifiable Components* by Scott A. Hissam, *Software Engineering for End-User Programmers* by Dr. Curtis Cook et al., *Competitiveness Versus Security* by Don O'Neill, and this month's online article, *Introducing TPAM: Test Process Assessment Model* by Dr. Yuri Chernak.

CrossTalk's Health Watch

Finally, as we discuss how to determine the health of an organization, you may be wondering how CrossTalk's health is at this time. As we reported in our February issue, we lost our funding due to the closure of the Air Force Computer Resources Support Improvement Program, our previous sponsor. I am pleased to report that we will continue publishing CrossTalk through fiscal year 2004. We continue to explore new approaches to producing the journal beyond that. Our goal is to keep CrossTalk healthy and thriving for many tomorrows. Please visit <www.stsc.hill.af.mil> for the latest on our future status.

Tracy L. Stauder
Publisher

Why Be Assessed to the Most Prevalent Standard in Use Today?

Robert Vickroy
ABS Quality Evaluations, Inc.

Initially conceived as a common, one-shoe-fits-all quality management system standard, ISO 9001 has developed to become the baseline concept inherited by many industry sector schemes and models. This article summarizes the several quality system models and outlines the critical factors that contribute to the successful implementation of an effective, robust, quality system.

ISO 9000 standards started in the late 1980s to promote standardization of trade. Early use of the ISO 9001 Standard was in the European Union countries, which influenced international trade by requiring registration for companies selling products worth more than 100,000 Euros, the then-evolving European Union currency. Since that time, the requirement to be registered has also been incorporated into many U.S. companies' bidding requirements. In the expanding global economy, domestic companies must now compete with foreign companies that are achieving ISO certifications to become more acceptable as suppliers. As a result, U.S. companies in foreign markets find ISO certification useful in assuring their foreign customers that they have fundamental quality processes.

Determining whether to be assessed to ISO 9001 depends a lot on what motivates a company's management. Typically, a company becomes ISO 9001 registered because it (1) is *required* to do so by the customer, (2) wishes to *reduce customer audits* by becoming registered to ISO 9001, (3) is more aggressive and feels it would be *more competitive*, believing customers would look more favorably on suppliers that are registered, (4) thinks producing a *quality product* would be more *cost effective* and that being ISO registered would improve quality, (5) wishes to *expand the capability of its business* by adopting broader or more in-depth quality models, (6) incorporates a combination thereof. As quality improvement is a journey and not a destination, companies are likely to evolve through several of these steps as they mature in the pursuit of quality.

As a result of reasons like these, there were 561,747 ISO 9000 certificates at the end of 2002 distributed throughout 159 countries, according to the International Organization for Standardization[1] in Switzerland, which owns ISO 9001 and other ISO-related standards. While the worldwide distribution of registrations changes constantly, a breakdown of the number of registrations by economic trading block in 2001 shows that approximately 50 percent were in European countries, 25 percent in Asian countries, and 9 percent in the United States, with a rapidly increasing number in smaller countries who wish to be suppliers to the larger countries.

> *"With many governments today retiring their local standards, ISO 9001 has served to simplify and standardize the definition of a quality management system for both the customer and the supplier."*

I am told during audits, and it is confirmed in the numerous ISO 9001 surveys reported, that the benefits of certification include having documented processes versus tribal knowledge; being trained; understanding why things are done to assist in achieving the company's goals and objectives; reducing costs due to scrap, rework, and delay; and overall buy-in by employees that results in improved customer focus and participation in continual improvement. Companies also find that the registrar's corrective action process in ISO 9001 is a valuable addition to their ongoing improvement program.

Users of the ISO 9001 standard had goals that were twofold: Reduce the customer's cost of auditing suppliers, and reduce the cost of conforming to and being audited by customers. The maze of standards significantly added to a company's cost of business and ability to compete. For example, a survey in the mid-90s by members of the ISO/International Electrotechnical Commission (IEC) Subcommittee SC7 U.S. Technical Advisory Group for Software found more than 500 standards worldwide for software alone. The numerous standards resulted in multiple audits of suppliers to conflicting requirements resulting in complicated and dissimilar quality management systems.

While auditing companies in the early '90s that had complicated quality systems, I asked how they had developed their quality management systems. The common response was that since each auditor who came through required something different, they incorporated those requirements into their quality system and it simply evolved.

With many governments today retiring their local standards, ISO 9001 has served to simplify and standardize the definition of a quality management system for both the customer and the supplier. Through ISO 9001's third-party auditing process, customer visits are typically reduced, which reduces costs while providing confidence that the company continues to operate in conformance with a registered quality system.

Companies that aggressively pursue different industry sectors often obtain registration to ISO 9001 as a baseline standard and may then adopt other standards associated with new business opportunities. ISO 9001 began as a one-shoe-fits-all quality system. However, several industrial sectors have documented additional specific requirements, referred to as sector schemes. These

® Capability Maturity Model, CMM, and CMMI are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

SM CMM Integration is a service mark of Carnegie Mellon University.

scheme's conform to ISO 9001 yet are required by the particular industry to demonstrate conformance to their elaboration on ISO 9001 requirements within the context of that industry's unique terminology, processes, and measurements. The result is incremental improvement based on the clauses of ISO 9001.

The software industry in the United Kingdom was the first to develop additional requirements documented in its TickIT Guide[2] and reflected by its notations on certificates: a tick, √ , (British for checkmark), followed by the letters IT (indicating Information Technology). This individualized sector scheme was later followed by other industries and documented in guides such as National Quality Assurance-1 for nuclear quality assurance; ISO 13485 for the U.S. Food and Drug Administration; Quality System 9000/Technical Specification 16949:2002 for the quality system in the United States and technical specification for the international automotive industry; Telecommunication Leadership 9000 for telecommunications; Aerospace Society 9000 for the aerospace standard; International Safety Management for marine international safety management; the FAA-iCMM from the U.S. Federal Aviation Administration; and ISO 9001 models for education, oil and gas, medical, and more.

Sector schemes are one way to improve relative to the performance maturity model in ISO 9004:2000 Appendix A.2. Continually improving by adding to and going beyond ISO 9001 with more in-depth quality principles in maturity models such as the Capability Maturity Model® (CMM®) not only indicates maturity to those models' paradigms, but also increases performance maturity relative to the ISO 9004 performance model.

Other process improvements include incorporating additional quality principles, enhancing metrics to achieve objectives with Six Sigma, pursuing broader excellence standards like Baldrige, or incorporating the new Space Systems – Risk Management[3] ISO 17666:2003 standard. ISO 9001 is flexible enough to allow a company to blend sector schemes or maturity model terminology and process detail that conform to or exceed ISO 9001 requirements when creating procedures. Conversely, a company implementing CMM/CMMI Level 3 processes finds many generally similar processes so that they only have to add relatively few unique clauses from ISO 9001 to also achieve a blended quality system. In either case, it is fundamental to begin with the end in mind in order to architect the building blocks (schemes) that will be blended into your quality management system over time to facilitate an orderly expansion of its features to suit the growth strategy of the company.

Having audited companies in many industries to ISO 9001 for many years, I have found that when companies truly apply ISO 9001, they mature from a mindset of being *forced* to do it to *wanting* to do it. The result is pride in quality products and an improved business environment achieved by truly applying the process.

Information Needed to Begin

The first thing people in a company need to know, and the biggest success factor, is that top management supports and provides the resources to implement the ISO 9001 quality management system. People do what top management takes an interest in, participates in, and can measure.

"Having audited companies in many industries to ISO 9001 for many years, I have found that, when companies truly apply ISO 9001, they mature from a mindset of being forced to do it to wanting to do it."

Companies should gather the information mentioned in this article, provide copies of the ISO 9001 standard (at least to key employees), and provide training to all employees on the ISO 9001 standard. They should also obtain the ISO 9000 glossary, ISO 9004 guidance document, and the free ISO guidance documents from the ISO/IEC Technical Committee 176 Subcommittee 2, found at <www.iso.org/iso/en/iso9000-14000/iso9000/transition.html>.

Any organization implementing ISO 9001 is encouraged to download this information so it understands the intention of the ISO 9001:2000 authors, and it correctly defines its quality management system – be skeptical of anyone who offers contrary advice. The documents include the following:

- "The Year 2000 Revisions of ISO 9001 and ISO 9004."
- "Transition Planning Guidance for ISO 9001:2000."
- "Guidance on Outsourced Processes."
- "Guidance on ISO 9001:2000 Clause 1.2 Application."
- "Guidance on the Documentation Requirements of ISO 9001:2000."
- "Guidance to the Terminology Used in ISO 9000:2000 Family of Standards."

Numerous Web sites offer help for ISO 9001 such as free quality manual templates found by searching Google. Such manuals are only a starting point and must be significantly enhanced to incorporate the processes, terminology, and tools used by a particular company. For example, according to ISO 9001 clause 4.2.2, the quality manual shall "include or reference procedures and describe the interaction between the company's processes."

Be aware that a diagram of the company's quality management system that simply copies Figure 1's process diagram from the ISO 9001:2000 standard instead of creating an actual process diagram of your company would not be generally acceptable to an ISO 9001 registrar. So be specific as sector schemes are in part the result of companies failing to voluntarily create industry-specific versus generic quality manuals and procedures in the early years of ISO 9001.

A thorough and honest analysis is a second success factor. An analysis of where the company complies with the standard, and where it needs to take action to establish compliance must be done to gain a realistic assessment of what needs to be accomplished.

A third success factor is to measure twice for every improvement. Companies are cautioned to proceed with a step-wise refinement of their quality system by establishing and measuring system performance before making improvements. Measure, then formalize what conforms, and add what is missing relative to the standard, then measure again. Then go on to reengineer processes and measure again. This establishes the data for the analysis required by ISO 9001 and substantiates the benefits of the quality system. Do not use ISO 9001 as a club to force unrelated pet improvement projects that were not accepted earlier; this is often a recipe for failure, or at least sig-

nificant delay in implementation.

Your Starting Contact

A company has to decide how quickly it wishes to achieve ISO 9001 registration, and what resources it has to apply to its effort to become registered. Another critical success factor is developing in-house competency by sending key personnel to an accredited ISO 9001 Lead Auditor class.

The ISO offers publications for help in getting started. For small businesses, the ISO also offers a free publication "ISO 9001 for Small Businesses[4]." If a company wishes to rapidly implement ISO 9001 and does not immediately have in-house resources, it may want to contact a consultant who is independent of the registrar. The ISO offers a free guide to selecting a consultant, "How to Choose a Competent Quality Management System Consultant[5]." Remember, the assessment requires that the company demonstrate the quality system is suitable and effective for its business.

The next step is to select a registrar. A resource to help you make that decision is "The ISO 9000 Handbook" [1]. *Quality Digest* magazine also offers an online list of registrars at <www.qualitydigest.com>. When selecting a registrar, begin with the end in mind. If you know the company intends to augment its quality management system with one or more of the ISO sector quality schemes or the CMM/CMMI, then consider a registrar who is also authorized to offer this added scope of service to ensure consistency.

How Much Time Will It Take?

The time it takes the company to prepare for the initial audit depends on where it is in the process of establishing a quality management system that conforms to the 31-page ISO 9001:2000 standard, and the degree of sophistication of its implementation.

The fundamental framework for estimating the number of audit days is defined by the International Accreditation Forum, Inc.[6] (IAF) in "IAF Guidance on the Application of ISO/IEC Guide 62:1966[7]." See the Annex 2 – Auditor Time, "Guide for Process to Determine Auditor Time For Initial Audit," and subsequent sections describing factors that may require more or less audit time. If a joint assessment is being performed to multiple standards, guidelines, or models, ensure that enough time is allowed to accomplish both successfully.

The process may start with a pre-assessment, which is an optional activity, preferably done by the person who will eventually be your auditor. IAF Guide 62 allows value-added assessments that can identify opportunities for improvement, but cannot result in recommendations or advice that would be considered consulting.

The typical process is an initial audit that is longer than subsequent surveillances, as the entire quality system of the company must be audited. An example estimate, drawn from IAF Guide 62 Annex 2, would be an audit of 276-425 employees in one location by two auditors for five days, adjusted per Guide 62.

Subsequent surveillance audits are semi-annual or annual, depending on the arrangements and confidence in the internal audit process of the company, and incrementally cover different clauses of the standard. An annual audit is typically twice as long as a semi-annual audit. After every audit, the registrar also verifies the audit report for conformance to its procedures. Overall, a typical registrar's contract is for three years, after which the current requirement is that the full quality system be re-audited to assure the overall system's continuing effectiveness has been maintained.

> *" ... remember the audit process is a journey; the auditor can add value in identifying opportunities for improvement as well as nonconformance for breakdowns or deviant evolution in the quality system that needs to be pointed out."*

What Is the Cost?

Each registrar must be contacted separately as it sets its own day rates, though market forces tend to make rates somewhat similar. The cost is typically determined for the three-year contract, which can be determined once the audit days and day rate have been agreed on, plus any pre-assessments that may be performed and the number of report reviews over the three-year period.

Some companies feel that with registration they have reached their destination (registration) and seek multi-site arrangements and bargain for price. However, remember the audit process is a journey; the auditor can add value in identifying opportunities for improvement as well as nonconformance for breakdowns or deviant evolution in the quality system that needs to be pointed out.

While companies have been known to perform a detailed cost analysis of their efforts, surveys of registered companies have typically shown that when measurements have been established as described above, net benefit can be demonstrated.

Audit Expectations: Before and After

Going into the audit, the company's quality personnel need to be sure of top management's commitment to finding and fixing any issues that may exist. Evidence is overwhelming that, if these issues are ignored or not addressed, they will resurface as even bigger problems later. This is a critical factor for now and the future. Top management must reinforce that identifying problems and opportunities for improvement is a fundamental goal of the quality system.

The company must expect to provide an escort for each auditor and have arranged a schedule with each auditor establishing that all processes and departments identified for audit can be accommodated in the time available. When the audit starts, the overall process is the following:

- Plan the audit with the lead auditor.
- Hold an opening meeting.
- Allow time for initial document review.
- Conduct numerous interviews.
- Request documents.
- Convene interim feedback sessions as appropriate.
- Allow time for the auditors to formulate results and audit reports.
- Hold a closing meeting to present the findings and the lead auditor's decision on whether a certificate can be granted.

Findings that are observations or non-systemic nonconformance are handled after the audit. Occasionally an initial assessment finds systemic failure to implement clauses of the ISO 9001 standard or the company's own proce-

dures. This will result in a reassessment to confirm completion of the missing clauses and a delay in issuing the certificate until implementation can be verified.

After the audit, the company must respond in writing to the audit findings (which may be nonconformance or observations), receive an acceptance of the response from the registrar, receive a certificate, and continue to take the committed corrective action to prevent reoccurrence of each finding.

A cycle of surveillance audits similar to the initial audit to verify continued compliance with selected clauses of ISO 9001 as described earlier is performed. The value-added audits add another dimension to the improvement process and help ensure the continued functioning of the quality system.

It is a natural expectation that the desire for continual improvement will cause findings to be resolved. Occasionally during surveillances, failure to implement effective corrective action may result in additional nonconformance; repeat nonconformance for the same finding over time may result in withdrawal of the registration certificate.

All registrars are required to provide a directory of currently registered companies. As the directory must be provided on request, the registered company will not only be listed in the registrar's directory but also in several other compilations of all registrars' directories available by subscription from publications such as *Quality System Update* or <www.qualitydigest.com>.

After your company becomes registered to ISO 9001, you should read "Publicizing Your ISO 9001:2000 or ISO 14001 Certification[8]" so you do not violate ISO restrictions. The ISO does not allow registered companies to use their symbol, often referred to as a *mark* in their advertising or literature.

Registrars often offer the Mark of Accreditors represented on the registration certificate given to the registered company and the Registrar's Mark to their clients for use in advertising or in their literature. The Registrar's Marks must be accompanied by the company's registration number so customers can verify the registration's validity. After registration, contact the registrar to obtain the marks as well as the restrictions on their use, which per the IAF Guide 62 prohibits using the mark on actual products.

Customers often periodically ask for copies of the current certificates held by a subcontractor to verify registration claims and gain confidence that the company has a quality system that is being audited by an independent third-party registrar's auditor. While many registered companies display their ISO 9001 and other certificates on their own Web site, it is more appropriate to verify that the certificate is current and represents the current scope of registration of the registered company through independent sources.

Independent verification of certificates by the customer is also necessary, as registrars have discovered fraudulent certificates. With more than 500,000 registered companies worldwide and with worldwide subcontracting and e-commerce, the customer must beware.

"Going into the audit, the company's quality personnel need to be sure of top management's commitment to finding and fixing any issues that may exist."

The registered company must be aware of the restrictions or charges its registrar places on the duplication of certificates, which can become expensive if it has several locations it wants to display the certificate, or if its customers ask for copies of certificates. Some registrar's directories are online and may display the certificate or permit the customer to print the certificate if the company is currently registered.

Summary

In summary, being assessed to the most prevalent standard in use today can establish the foundation for quality in a company, achieve more immediate benefits to the business, establish universal recognition of a standardized quality management system, and lay the groundwork for continued improvement by expanding to incorporate other quality techniques, sector schemes, and quality models.◆

Reference

1. Peach, Robert W. The ISO 9000 Handbook Fourth Edition. QSU Publishing Company, 11 Oct. 2002.

Notes

1. See <www.iso.org/iso/en/commcentre/pressreleases/2003/Ref864.html>.
2. See <www.tickit.org/international.htm>.
3. See <www.iso.org/iso/en/isoonline.frontpage>.
4. See <www.iso.org/iso/en/iso9000-14000/basics/basics9000/basics9000_4.html>.
5. See <www.iso.org/iso/en/commcentre/isobulletin/articles/2001/pdf/qmconsultant0112.pdf>.
6. See <www.iaf.nu> for documentation.
7. See <www.accreditationforum.com/guidance.asp> under Guidance Documents (GD Series).
8. See <www.iso.org/iso/en/iso9000-14000/publicizing/index.html>.

About the Author

Robert Vickroy is a senior auditor for ABS Quality Evaluations, Inc., an ISO registrar for ISO 9001 and sector schemes and for ISO 14001, and a Software Engineering Institute Transition Partner. Previously Vickroy worked for IBM for 25 years developing systems and automating processes in all areas of IBM engineering, manufacturing, and in software development as programmer, analyst, and manager of software development organizations. As an auditor for TickIT, ISO 9001, ISO 14001 and TL9000, and as a SEI authorized Capability Maturity Model® (CMM®) Assessor and CMM Integration℠ Appraiser, Vickroy has assessed more than 300 quality management systems, specializing in assessing jointly implemented systems. Additionally, he achieved the following certifications: ASQ-CQA, ICCP-CDP, EDPA-ISACA and ISC2 security auditor, and he is a trained NQA-1 auditor. Vickroy has a Bachelor of Arts in computer science, a Bachelor of Arts in accounting and economics, and a Master of Science.

ABS Quality Evaluations, Inc.
16855 Northchase DR
Houston, TX 77060
Phone: (281) 877-6485
Fax: (281) 877-6001
E-mail: bvickroy@eagle.org

CMMI Myths and Realities

Lauren Heinz
Software Engineering Institute

This article seeks to deconstruct several myths circulating through the software engineering community about upgrading from the Capability Maturity Model® for Software (SW-CMM®) to the CMM Integration℠. Early adopters and model experts share advice for how your organization can make the upgrade.

Many myths are often heard in discussions about upgrading from the Capability Maturity Model® for Software (SW-CMM®) to the CMM Integration℠ (CMMI®):

1. The CMMI is too big and complex.
2. A CMMI appraisal takes longer and costs more than one for SW-CMM.
3. The CMMI is only for large organizations.
4. The CMMI is only for enterprise-wide process improvement.

However, those who have been using the new set of models, appraisal method, and training materials contend that making the upgrade to the CMMI Product Suite is not only easier than it looks, but also well worth it.

"For organizations already operating at a high SW-CMM maturity level, the process of adopting CMMI is very straightforward," said Sarah Bengzon, an associate partner at Accenture, a leading management consulting and technology services organization. "People think that with CMMI everything is new and that the process is too complex to undertake. But at Accenture, we have always been doing things this way. If anything, CMMI validates the best practices we already had in place."

In fact, Accenture's USA Government Operating Unit, which is an early adopter of the CMMI Product Suite, attained CMMI Maturity Level 3 just eight months after making the upgrade from the SW-CMM model. "CMMI enforces tying project objectives to organizational objectives, which is not only a good thing to do, but a bad thing not to do," Bengzon said. "CMMI shows you exactly what you should be doing to improve your quality processes."

Accenture's group is just one of hundreds making the upgrade to the CMMI worldwide. To date, more than 16,000 people have attended an Introduction to CMMI course offered by the Software Engineering Institute (SEI℠) and its transition partners, more than 230 instructors have been trained to teach the introductory course, and more than 290 individuals have become authorized Standard CMMI Appraisal Method for Process Improvement (SCAMPI℠) Lead Appraisers. "Initial acceptance of CMMI seems to be much faster than it was for SW-CMM," said Bill Peterson, director of the Software Engineering Process Management Program at the SEI.

While some myths from the earlier development and piloting days of the CMMI models are still circulating, Bengzon and others are proving that these misconceptions are easy to clear up with a little guidance from the experts.

> *"For organizations already operating at a high SW-CMM maturity level, the process of adopting CMMI is very straightforward ... If anything, CMMI validates the best practices we already had in place."*

CMMI Myths

1. CMMI Is Too Big and Complex

For more than 10 years, the SW-CMM model has been the global, de facto standard for appraising and improving software processes. As organizations came to know and experience the value of the SW-CMM model and other capability maturity models, these organizations sought to expand the use of the capability maturity model concept beyond its initially defined scope. This evolution of the capability maturity model concept naturally grew into the development of the CMMI Product Suite. Its purpose is to provide guidance for an organization to improve its processes and its ability to manage the development, acquisition, and maintenance of products and services. The CMMI Product Suite places proven practices into a structure that helps an organization appraise its organizational maturity and process capability, establish priorities for improvement, and guide the implementation of these improvements.

However, at 700-plus pages each, the CMMI models can seem a bit daunting.

Roger Bate, principal architect of the CMMI Product Suite, said the models are so lengthy because they provide comprehensive guidance and details. "It's similar to an encyclopedia," he said. "There are a lot of subjects in there that you'll never need to look up, but they're there so they can be available to everyone when and if they need them."

The most obvious additions to the models are related to integrated product and process development (IPPD), which now includes two additional goals and three new process areas (PAs) called Integrated Teaming, Organizational Environment for Integration, and Integrated Supplier Management. Best practices covering risk management were also enhanced. In addition, the SW-CMM's single Software Product Engineering key process area was expanded into five, more comprehensive PAs in the CMMI Product Suite. A Measurement and Analysis PA at maturity Level 2 and a Decision Analysis and Resolution PA at maturity Level 3 were also added to the models.

"But don't let the page count throw you," Bate said. He recommended three ways that an organization can address this myth:

1. **Select the right model.** There are several CMMI models to choose from, including CMMI for Software Engineering, CMMI for Systems and Software Engineering (SE/SW), CMMI SE/SW with IPPD, and CMMI SE/SW with IPPD and Supplier Sourcing. Once you select a model, tailor it to fit your organization's needs.
2. **Do not try to implement an entire model at once.** "Select those parts that are most applicable and will have the

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

biggest payoff at the first stage of process improvement," Bate said. "Get at those things that are most important: improving quality, predicting costs and schedules, and reducing time to market. Develop a base from which you can move forward."

3. **Follow the practices that make the most sense for your organization.** "You can pick and choose or substitute your own processes as long as they meet the overall goals. Every sub-practice does not need to be implemented. They are informational guides, not requirements," he said.

Additionally, an organization can further tailor its adoption of a model by selecting the staged or continuous representation.

Organizations new to process improvement tend to prefer a staged approach, which 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 model, are more 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 which process areas it wants to address and to what degree. Instead of maturity levels, capability levels (0-5) are used to measure improvement against the best practices of a single process area. Generally, an organization that does not want a maturity level to help it compete with other businesses might select continuous.

Although there are several small differences, process experts agree that both representations contain nearly identical information. Either one will help an organization improve its products, projects, and processes.

2. A CMMI Appraisal Takes Longer and Costs More Than One for SW-CMM

The SCAMPI Class A appraisal method, which is used to appraise an organization's use of CMMI best practices, is designed as an Appraisal Requirement for CMMI (ARC) Class A appraisal method. It is intended for use where the highest confidence and accuracy is desired on the part of the appraisal sponsor. David Kitson, principal architect of the SEI appraisal methods, said he and the SEI Appraisal Program team have seen a number of SCAMPI Class A appraisals performed and have been very happy with the results.

After hearing three years ago from CMMI early adopters that the SCAMPI method was often taking 150 or more hours for a maturity Level 3 appraisal, Kitson and a team from government, industry, and the SEI adopted a stretch goal: Maturity Level 3 appraisals would take no longer than 100 hours on site. Since the first round of SCAMPI Lead Appraiser℠ Training in April 2002, Kitson said, one defense contractor has reported conducting its maturity Level 3 SCAMPI appraisal in just 60 hours.

"We are seeing in practice the realization of the benefits we expected SCAMPI would provide," Kitson said. "The organizations that are reaping the maximum benefits that SCAMPI offers are the ones that are taking the time to make genuine improvements in their processes and to treat process improvement just as they would any other project they undertake."

Additionally, the SEI has developed two alternatives to a SCAMPI Class A appraisal: SCAMPI Class B and SCAMPI Class C.

> *" ... the cost of getting the new processes adopted and used by the intended scope of the organization is typically much less for a small company, and the deployment can go faster."*

Although neither method can be used to produce a maturity level rating, both can be used to help organizations gauge the state of their process improvement and uncover process strengths and weaknesses.

"These methods can take less time, depending on the scope of the appraisal, and provide much more flexibility," said Jack Ferguson, who leads the SEI appraisal program. "The Class B method," Ferguson explained, "is slightly more rigorous than the Class C method. It requires a minimum-sized team to perform the appraisal and a corroboration of appraisal artifacts through interviews or other methods that demonstrate the practices are being performed. Class C can be done entirely with interviews or with document and artifact review.

"When you are looking for a rating, it is necessary to use Class A," Ferguson said. "But if you're doing the appraisal to help yourself, or you want to give upper management a sense of where things stand, both B and C are good options."

3. CMMI Is Only for Large Organizations

Although the CMMI models were developed in part to help larger organizations tackle complex issues across multiple disciplines, they can be tailored to meet the needs of smaller companies and organizations.

The SEI and the Army's Software Engineering Directorate at Redstone Arsenal have partnered for a pilot study to implement a subset of CMMI process areas at two small companies in the Huntsville, Ala., region. The focus of the study is to enable better understanding of the enablers and barriers to CMMI adoption in the small company environment, while demonstrating business benefit to the companies involved.

"With the Huntsville pilots, our experience is that you use CMMI differently than you might in a larger organization," said Suzanne Garcia, a member of the piloting team at the SEI. "Because of the limited resources in a small company for supporting process infrastructure, we took the approach of analyzing the business issues that were giving the companies problems, and using the related process areas and generic practices to help them solve those problems."

Garcia and her teammates identified three major cost areas for using the CMMI in most organizations: (1) the periodic cost of conducting an appraisal, (2) the cost of establishing and maintaining a process improvement infrastructure, and (3) the cost of deploying new processes throughout the organization.

"A large company has an advantage in the first two cost areas, because the cost of appraisal and the cost of infrastructure will be a smaller percentage of their overall revenue than for a small company," she said. "However, the cost of getting the new processes adopted and used by the intended scope of the organization is typically much less for a small company, and the deployment can go faster. If the small company can find ways to reduce the cost of the appraisal and infrastructure, they actually may have an overall advantage in getting business benefit from using the CMMI over a large company."

4. CMMI Is Only for Enterprise-Wide Process Improvement

In 2003, the SEI launched an Interpretive Guidance project to collect information

about how the CMMI is being utilized by software, information technology, and information systems organizations, and to identify problems these organizations may have as they adopt the CMMI.

Mary Beth Chrissis, project manager of the Interpretive Guidance project, said the project was formed to respond to organizations that were interested in implementing only the software engineering best practices. "These 'software-only' organizations, we were told, were having some difficulty applying CMMI in their environments," she said. "This project set about collecting information to find out what these problems were."

The project gathered information using various methods, including an online survey and meetings at process improvement events. A preliminary report, published in late 2003, summarized the data gathered. The results were surprising[1].

"We expected to see patterns that would help us identify problems with the CMMI models that were causing specific trouble for software-only organizations. Instead, we found that these organizations were experiencing very few problems with CMMI," she said. "If anything, this project has validated that CMMI models meet the needs of software-only organizations just as well as those pursuing enterprise-wide process improvement."

Conclusion

The CMMI Product Suite is a set of products that enable users to improve their product and service development and maintenance processes. These products include a set of CMMI models, the SCAMPI appraisal method, and the CMMI training program. Hundreds of organizations are currently using the CMMI Product Suite and sharing their experiences with the SEI. While some misconceptions from the early development and piloting days of the CMMI project are still circulating, those with experience using the product suite are helping to resolve and dismiss many of these initial concerns.◆

Notes

1. For more information, please see: <www.sei.cmu.edu/publications/documents/03.reports/03sr007.html> and <www.sei.cmu.edu/ publications/documents/03.reports/03sr009.html>.

About the Author

Lauren Heinz is a technical writer/editor at the Software Engineering Institute.

Software Engineering Institute
Pittsburgh, PA 15213-3890
Phone: (412) 268-1750
Fax: (412) 268-5758
E-mail: lheinz@sei.cmu.edu

Web Sites

ISO

www.iso.ch/iso/en/ISOOnline.frontpage

The ISO is a network of the national standards institutes of 148 countries, on the basis of one member per country, with a central secretariat in Geneva, Switzerland that coordinates the system. The ISO is the world's largest developer of technical standards, including ISO 9000, ISO 14000, and more than 14,000 international standards for business, government, and society.

Capability Maturity Model Integration

www.sei.cmu.edu/cmmi

The Software Engineering Institute hosts the Capability Maturity Model® Integration (CMMI®) Web site. It features general information about the CMMI, the latest models, how to get training, help with adoption, information about appraisals, and background information about the CMMI project as well as tips and information for the newcomer.

iSix Sigma

www.isixsigma.com

iSixSigma is a free information resource created to meet the needs of business professionals in search of proven methodologies for improving process efficiency, implementing data-driven decision making, and focusing on customer needs. The site offers comprehensive information, unique tools, checklists, calculators, and in-depth editorial and personalized advice to help quality and management professionals implement Six Sigma quickly and successfully into their organizations.

Software Technology Support Center

www.stsc.hill.af.mil

The Software Technology Support Center is an Air Force organization established to help other U.S. government organizations identify, evaluate, and adopt technologies to improve the quality of their software products, efficiency in producing them, and to accurately predict the cost and schedule of their delivery.

The Quality Assurance Institute

www.qaiusa.com

The Quality Assurance Institute (QAI) is dedicated to partnering with the enterprise-wide information quality profession. QAI is an international organization consisting of member companies in search of effective methods for detection-software quality control and prevention-software quality assurance.

Software Productivity Consortium

www.software.org

The Software Productivity Consortium (SPC) is a nonprofit partnership of industry, government, and academia. The SPC develops processes, methods, tools, and supporting services to help members and affiliates build high-quality, component-based systems, and continuously advance their systems and software engineering maturity pursuant to the guidelines of the major process and quality frameworks.

Software Program Managers Network

www.spmn.com

The mission of the Software Program Managers Network (SPMN) is to identify proven industry and government software best practices and convey them to managers of large-scale software-intensive acquisition programs. The SPMN enables program managers to achieve project success and deliver quality systems on schedule and on budget. More than 200 Department of Defense programs have benefited directly from SPMN expertise, consulting, and assessments.

There Is More to Process Improvement Than Just CMM

Dr. Linda Ibrahim
Federal Aviation Administration

Joan Weszka
Lockheed Martin Corporation

There are many models and standards that provide guidance for improving software, systems, and other organizational processes. The scope of these standards and models is more extensive than the Capability Maturity Model® for Software (SW-CMM®), and CMM Integration℠. This article describes approaches taken at the Federal Aviation Administration and at Lockheed Martin to assure process improvement meets comprehensive needs across these enterprises.

Fueled by the premise that improving products is predicated on improving processes used to develop and deploy them, early process improvement efforts based on the Capability Maturity Model® (CMM®) were focused on software systems. Documented cases of software systems fraught with problems underscored the need for scrutinizing software engineering processes against an industry-standard model compiled from proven best practices.

A plethora of benefits have been attributed to using the CMM for Software (SW-CMM)[1] across a broad spectrum of areas directly related to business growth and success. Such benefits include improvements in quality (measured in terms of defect reduction or earlier detection), productivity, cost, and schedule. Due to the scope of the model used, these benefits focused on the software aspects of system development, since improvements were typically constrained to software engineering processes and those directly supporting them.

Successful software process improvement spawned the development of models focused on other disciplines, including systems engineering and work force management. Each new model was earmarked for use across a subset of an organization, e.g., the systems engineering or software engineering organizational elements. The resulting stovepiped approach to process improvement resulted in inefficiencies caused by a different model for each discipline, and often inattention to integrated process improvement. However, as capability maturity model use extended across an enterprise, the benefits also accrued in those areas where process improvement ensued.

In 1998, an industry and government need surfaced for an integrated maturity model to achieve efficiency and effectiveness of processes and process improvement in a multidisciplinary environment. Earlier work by the Federal Aviation Administration (FAA), as described in this article, demonstrated a proof of concept for an integrated model. The 1998 industry/government effort led to creation of the CMM Integration℠ (CMMI®); CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing (CMMI-SE/SW/IPPD/SS) V1.1[1], and several variations with more limited scope.

> *"Successful software process improvement spawned the development of models focused on other disciplines, including systems engineering and work force management."*

Source models used to create CMMI were the SW-CMM, Systems Engineering CMM (SE-CMM) Electronic Industries Alliance/Interim Standard (EIA/IS) 731[2], and the Integrated Product Development CMM[1]. The CMMI model, with its focus on systems engineering, software engineering, integrated process and product development, and supplier sourcing had a broader scope than its predecessor's single-discipline models. However, it still lacked comprehensive coverage of broader enterprise processes.

Approaches to Working Beyond the CMMI

Complex enterprises like the FAA and Lockheed Martin engage in engineering activities; operations; acquisition; supply; strategic and portfolio management; financial management; human resource management; and a host of technical, management, and support functions to operate their business. A number of these enterprise processes are not currently covered by the CMMI. However, there are multiple industry standards and models that can provide additional process improvement guidance to bridge the gap.

Both Lockheed Martin and the FAA have developed approaches to address process improvement needs that extend beyond CMMI and have incorporated practices and guidance from additional sources into their process improvement programs. The FAA and Lockheed Martin approaches differ, however, as described below.

The FAA Approach

The FAA's approach is to integrate existing models and standards into a single process improvement framework that can be used by any organization to guide process improvement within the scope of the model. The framework was developed with government and industry participation and is not FAA-specific. The framework has been designed to be flexible, with process areas used selectively according to the business needs of the implementing organization. The framework's scope continues to expand as explained below.

First Integrate Capability Maturity Models

The first problem the FAA faced was the concurrent use of multiple, single-discipline capability maturity models. Prior to 1997, the FAA was using the SW-CMM, the SE-CMM[1], and the Software Acquisition CMM (SA-CMM)[1]. Each model provided guidance for different aspects of the FAA's work, which was very useful. In some instances, however, all three models provided guidance for the same work, performed by the same integrated team, which was confusing since the three models have different architectures and approaches, and use different, sometimes inconsistent, terminol-

ogy. It was inefficient and ineffective to use the three models concurrently.

To solve this multiple-capability maturity model problem, the FAA integrated the SW-CMM, SE-CMM, and SA-CMM into a single integrated capability maturity model known as FAA integrated Capability Maturity Model (FAA-iCMM, or simply iCMM). The iCMM v1.0 was released in 1997 and as the first major integrated capability maturity model, it demonstrated that it was possible to integrate capability maturity models of different structures and scopes into a single model capturing all the principles and practices of the sources, using a single *continuous with staging* representation that includes both capability and maturity levels [1].

Next Integrate Beyond Capability Maturity Models

The iCMM rapidly became the predominant framework for capability maturity model-based improvement in the FAA, with programs and organizations making major strides in integrated process improvement. Yet there remained other critical software-related processes as well as broad enterprise processes that were not included in iCMM v1.0. Furthermore, there were non-capability maturity model standards that were of interest to stakeholders such as ISO 9001:2000 Quality Management Systems[3] and Malcolm Baldrige National Quality Award criteria.

Since the concepts and approach of the iCMM were becoming institutionalized, the following question emerged: Can other standards and models besides capability maturity models be incorporated into the same framework? The FAA rose to this challenge and the iCMM was revised and expanded to update software and systems engineering guidance to the latest standards, to expand iCMM scope to address the full software/systems life cycle, and to address enterprise management. A total of 10 standards and models[4] were integrated into iCMM v2.0, which was released in 2001 [2].

Each source integrated into iCMM v2.0 provided valuable insights and contributed to the content, comprehensiveness, and cohesiveness of the model. The following are some examples:

- **Life-Cycle Coverage.** Two source standards specifically intended to establish common frameworks for the life cycle were integrated: the ISO/International Electrotechnical Commission (IEC) 12207 Standard for Information Technology – Software Life-Cycle Processes[3], and the ISO/IEC 15288 System Engineering – System Life Cycle Processes[3]. These contributed to new process areas extending iCMM life-cycle coverage beyond development and maintenance to include deployment, transition, disposal, operation, and operational support.
- **Acquisition and Supply.** Three source standards (ISO/IEC 15288, ISO/IEC 12207, and ISO/IEC 15504 Information Technology – Software Process Assessment[3]) provide guidance for both acquisition and supply activities performed in an enterprise.
- **Strategic Management, Business Results, and Performance Measurement.** Strategic management guidance is provided in Baldrige, ISO/IEC 15288, ISO/IEC 15504, and ISO 9001, contributing to a new process area for enterprise management. The iCMM reflects a strong emphasis on performance management and business results that are fundamental in Baldrige, and the importance of measurement is strongly reinforced in both Baldrige and ISO 9001.
- **Quality Management.** ISO 9001 influenced the inclusion of prevention and root cause analysis as a natural part of quality assurance in the iCMM, as well as the need to determine customer satisfaction.
- **Maturity and Capability Levels.** The CMM and capability maturity model-type sources for iCMM (CMMI, EIA/IS 731, SA-CMM, SW-CMM and SE-CMM) served to consolidate definitions of maturity levels (across the staged models) and capability levels (across the continuous models, with ISO/IEC 15504 also providing input regarding capability levels and generic practices).

For details showing how each source contributed to iCMM v2.0 at the practice level, see [3].

The iCMM continues to evolve with recent government/industry projects focusing on synthesizing and harmonizing standards-based best practices in safety and security assurance for use with both iCMM and CMMI [4], and in developing guidance for use of a common process improvement framework in the context of developing and using an enterprise architecture. Accompanying the iCMM are public training courses and a variety of appraisal methods as described in this article in the section Appraising Beyond Capability Maturity Models.

> *" ... the FAA is piloting the Single Appraisal, Multiple Certification idea in collaboration with ISO 9001 auditors, whereby a single appraisal-audit process can result in both ISO 9001 certification and the iCMM appraisal results."*

Lockheed Martin Approach

Lockheed Martin has had a long history of involvement in model-based process improvement and demonstration of high maturity using the SW-CMM, the EIA/IS 731, the CMMI, and other models. In 1998, an internal corporate study, "The Elements of Success," focused on program performance and reaffirmed process performance as a critical success factor in program performance. Subsequent to the study came the realization that integration of single-discipline processes across the organization was not guaranteed without a mechanism for measuring enterprise process integration. As a result, Lockheed Martin created the Integrated Engineering Process (LM-IEP) Standard project to establish an integrated engineering process standard, a supporting infrastructure, and a measurement framework that enables collaborative, integrated, engineering and enterprise environments.

The LM-IEP Standard [5] provides a set of process integration requirements to be satisfied by each business unit's organizational standard process and related command media. The purpose is to create a concise, non-overlapping set of normative requirements applicable across a broader segment of the enterprise than covered by any individual industry standard or model already in use across the corporation. The integrated standard also allows for more efficient standards compliance, given the degree of overlap of several of the source documents. A Lockheed Martin corporate policy requires each business unit to conform to the standard, with application guidelines and timetables.

The LM-IEP Standard Revision 2.0 synthesizes requirements from CMMI-SE/SW/IPPD/SS V1.1, American National Standards Institute/EIA 632 Processes for Engineering a System[2], ISO 9001:2000, ISO/IEC 12207, ISO/IEC 15288, Institute of Electrical and Electronic Engineers 1220 Standard for the Application and Management of the Systems Engineering Process [6], and an internal Lockheed Martin standard for hardware engineering. In 2004, AS9100 (Quality Systems – Aerospace – Model for Quality Assurance in Design, Development, Production, Installation, and Servicing) is being added, along with additional details on process architecture conformance.

Complementing the LM-IEP Standard is a comprehensive product suite, including training, a corporate-wide Process Asset Library, integrated measurement and risk management guides, and an appraisal method as described in the section below, Appraising Beyond Capability Maturity Models.

Benefits from implementing the SW-CMM had been previously demonstrated by Lockheed Martin business units; there was high expectation that these benefits could be multiplied by deploying process requirements across a broader segment of the enterprise than required by stove-piped capability models already in use. Additional benefits accrue from using integrated processes and teams, founded on IPPD principles. Conformance to the LM-IEP Standard also provides a shared vision for integrated processes across the corporation and facilitates sharing work across business units.

Appraising Beyond Capability Maturity Models

Traditionally each maturity model has had its own appraisal methods issued as part of the product suite. For the SW-CMM, the methods include the CMM-Based Appraisal for Internal Process Improvement (CBA-IPI)[1] and the Software Capability Evaluation (SCE)[1]; for the CMMI, the method is the Standard CMMI Appraisal Method for Process Improvement (SCAMPI)[1]. Each of these methods complies with a defined set of appraisal requirements: the CMM Appraisal Framework[1] for the CBA-IPI and the SCE, and the Appraisal Requirements for CMMI (ARC)[1] for the SCAMPI.

It is important to distinguish between appraisal methods and the reference models against which they appraise. Appraisal methods should be generic and applicable to any reference models that align with basic architectural structures used during appraisal such as goals (outcomes) and practices (activities) expected to be performed to achieve goals. The specific content of the reference model is not relevant as far as applicability of an appraisal method is concerned. Both Lockheed Martin and the FAA have developed appraisal methods that can be used to appraise processes in areas that extend beyond the SW-CMM and CMMI.

Lockheed Martin Continuous Appraisal Method

Lockheed Martin initially developed the Continuous Appraisal Method (CAM) [7] for use with EIA/IS 731, but the method is equally applicable to CMMI as well as other models with analogous architectures. To date, the CAM has been deployed extensively across the corporation with CMMI, and the method has been shown via a pilot to be well suited for appraisal against the extended process requirements in the LM-IEP Standard.

Having extensive experience with CMM appraisals using the CBA-IPI, Lockheed Martin developed the CAM with a vision of a new paradigm for process appraisal and improvement. The CAM differs from a traditional formal appraisal approach in its focus on appraising incrementally, over a period of nine to 12 months, with an opportunity to correct weaknesses documented during the appraisal, and have improved processes reappraised.

After identified weaknesses have been addressed, typically incrementally during the course of the appraisal, the CAM Maintenance Review is scheduled. This Review acts as a checkpoint to assure model compliance and process fidelity, i.e., that no backsliding occurred during the course of the appraisal. Limiting the overall appraisal period to a maximum of one year provides a boundary on the timeframe within which the organization must address weaknesses related to its target profile in order to achieve the desired rating.

The CAM was designed as a rigorous appraisal method, intended to satisfy all of the ARC Class A requirements. Additional design drivers for CAM included reducing appraisal cost; interleaving appraisal with process improvement in an open, penalty-free environment; minimizing appraisal disruption; and facilitating institutionalization.

The CAM reduces appraisal cost by minimizing appraisal preparation efforts, beginning with eliminating the need for preparing an extensive hardcopy objective evidence library. The ability to address weaknesses during the course of the appraisal also eliminates the need for multiple informal assessments to ensure that all of the practices/goals in the appraisal scope are in compliance before CAM begins. Extensive preparation of appraisal participants is unnecessary since CAM allows for explanation and/or clarification of practice interpretation during the course of the interviews, and there is no risk of failing the appraisal if a weakness is uncovered during an interview.

The CAM's interleaving of process appraisal with improvement (fixing weaknesses) allows for a timely feedback loop where practitioners get confirmation from the appraisal team that improvements resulted in model compliance. This approach also promotes shorter cycles of continuous process improvement as opposed to longer periods of process definition and rollout followed by extended periods of appraisal preparation and appraisal. Furthermore, there is no fear that failure to comply with a single goal/practice could result in missing achievement of the appraisal objective (e.g., a process maturity/capability level goal). As a result, CAM participants are more readily inclined to volunteer areas where improvement is warranted.

Appraisal disruption is minimized using CAM since the extended appraisal duration provides ample opportunity for scheduling around project and organizational milestones. In the case of a traditional two- or three-week formal appraisal, the impact on projects, as well as the organization, can be significant.

Although CAM initially focuses on a set of representative programs, the method promotes institutionalization across the organization by providing a mechanism for appraising additional projects following the initial appraisal. After the maintenance review, additional cycles of project appraisals can continue until all programs in the organization have been appraised. During each project appraisal cycle, CAM requires indicators of at least three months of process implementation as evidence that the process has been institutionalized.

FAA Integration of Appraisal Methods

Just as the FAA chose an integration approach for development of the iCMM reference model, it similarly integrated various appraisal methods for use in a variety of process improvement contexts. The evolution of the FAA-iCMM Appraisal Method (FAM) [8] has mir-

rored the evolution of the model.

Integrate Various Capability Maturity Model-Based Appraisal Methods

The FAM integrates a variety of appraisal approaches, offering six methods and variations: Full Internal, Full External, Questionnaire-Based, Interview-Based, Document-Intensive, and Facilitated Discussion. These methods draw upon various capability maturity model-based appraisal methods including the CBA-IPI, the SE-CMM Appraisal Method[1], the SCE, and the Interim Profile[1]. In addition, the FAM formally describes methods based on document review and facilitated discussion self-appraisal. It is also possible to use the SCAMPI with the iCMM since the iCMM and the CMMI architectures are compatible. Similarly, the FAM variations are being used in safety and security assurance pilot appraisals that appraise organizational processes against both the iCMM and the CMMI.

Provide Multiple Results With a Single Appraisal

Improvements realized when using the iCMM simultaneously yield improvements against all its source standards and models. For example, achieving maturity Level 2 on the iCMM aligns with achieving maturity Level 2 on all its staged sources, including the CMMI, the SA-CMM, and the SW-CMM. But what happens when going beyond capability maturity models?

For example, organizations pursuing iCMM-based process improvement might also have a business objective to achieve ISO 9001 certification; organizations that are already ISO 9001 certified might have additional business goals that iCMM can support. Such simultaneous improvements can be accomplished efficiently with an integrated model; it is important to provide explicit guidance regarding these needs [9]. To implement and demonstrate this concept, the FAA is piloting the Single Appraisal, Multiple Certification idea in collaboration with ISO 9001 auditors, whereby a single appraisal-audit process can result in both ISO 9001 certification and the iCMM appraisal results.

Appraise More Than Capability

The iCMM also has an appraisal method designed to measure the usefulness and cost effectiveness of process performance results [10]. This method builds on generic attribute concepts introduced in the EIA/IS 731, and encompasses similar ideas found in ISO 9000 and ISO/IEC 15504. It focuses on performance results rather than capability.

Experiences

Both the FAA and Lockheed Martin have been implementing process improvement – beyond CMMI – for several years with resulting lessons learned.

For Lockheed Martin, the boldness of undertaking a corporate-wide LM-IEP Engineering Excellence Program whose scope was broader than any single model or standard is attributed to enlightened executive leadership that recognizes the business value of integrated process improvement across an enterprise. However, defining the process improvement agenda for the corporation based on the LM-IEP standard was no small feat. Specifically, synthesizing requirements from a diverse set of standards and models, many overlapping and written at varying levels of detail, was a difficult task that required expert knowledge of the source documents being synthesized.

A particular challenge was the objective of reducing the number of requirements in the LM-IEP standard to be significantly less than the composite number in the source documents. Furthermore, meticulous traceability of each requirement in the LM-IEP standard to its source was required to provide implementers with insight and informative references to facilitate understanding and interpretation.

For the FAA, using a single, flexible enterprise process improvement framework has paid off. Integrated iCMM-based process improvement has fostered shared improvement goals, a common improvement approach, and vertical and horizontal collaboration across disciplines, organizational lines, and the complete product or service life cycle. It enables organizations to focus improvement efforts on those parts of the iCMM that align with their business needs, and the model scope incorporates the business needs across a broad segment of the enterprise. The FAA's variety of appraisal methods has also facilitated improvement efforts.

A critical success factor in developing an enterprise improvement model is to recognize, incorporate, and integrate the principles and practices of international and national standards and performance-excellence criteria, while providing robust traceability to those sources.

Recommendations

Based on these experiences, the FAA and Lockheed Martin recognize the value of an integrated enterprise improvement framework to guide process improvement. Such a framework should be designed for flexible use across an enterprise, and should draw together widely recognized standards and approaches.

In future releases, the scope of the CMMI framework could be extended beyond engineering development and maintenance to address broader enterprise needs. For example, future extensions could include the following:

- Broader life-cycle coverage, e.g., deployment, transition, disposal, and operations.
- Broader enterprise coverage, e.g., acquisition, hardware engineering, finance, strategic management, work force management, information management, and the work environment.
- Mechanisms for adding specialty areas, e.g., safety and security.

In expanding the CMMI model scope, practices from international and national standards as well as other recognized best practices should be incorporated, as appropriate, with full traceability to sources. In addition to the model, the SCAMPI needs to be broadened to address incremental, delta, and multiple-certificate appraisals to meet user needs for efficient, effective appraisals in a variety of user modes and circumstances. A variety of successful methods should be considered for synthesis in developing new appraisal approaches.

Future releases of the CMMI Product Suite afford the opportunity to address broader, enterprise-level needs of organizations interested in realizing process improvement benefits across additional segments of their business.◆

References

1. Ibrahim, Linda, et al. The Federal Aviation Administration Integrated Capability Maturity Model (FAA-iCMM), v1.0. Washington, D.C.: Federal Aviation Administration, Nov. 1997 <www.faa.gov/ipg>.
2. Ibrahim, Linda. et al. The Federal Aviation Administration Integrated Capability Maturity Model (FAA-iCMM), v2.0. Washington, D.C.: Federal Aviation Administration, Sept. 2001 <www.faa.gov/ipg>.
3. Ibrahim, Linda, et al. Mapping Table Supplement to the FAA-iCMM v2.0. Washington, D.C.: Federal Aviation Administration, Oct. 2001 <www.faa.gov/ipg>.
4. Ibrahim, Linda, and Joe Jarzombek. "Safety and Security Extensions to Integrated CMMs." Software Engineering Process Group Conference 2004, Orlando, FL, Mar. 2004 <www.faa.gov/ipg>.
5. Lockheed Martin. Lockheed Martin Integrated Engineering Process (LM-

IEP) Standard, EPI 280-01 Revision 2.0. 3 Jan. 2003.

6. Institute of Electrical and Electronics Engineers, Inc. IEEE Std. 1220-1998, Standard for the Application and Management of the Systems Engineering Process. New York: 1998 <www.ieee.org>.
7. Carr, Marvin, and W. Neil Crowder. "Continuous Appraisal Method (CAM) … A New Paradigm for Benchmarking Process Maturity." Proc. of the Tenth Annual International Symposium of the International Council on Systems Engineering. Minneapolis, MN, July 2000.
8. Ibrahim, Linda, et al. The Federal Aviation Administration Integrated Capability Maturity Model (FAA-iCMM) Appraisal Method (FAM), v1.0. Washington, D.C.: Federal Aviation Administration, Apr. 1999 <www.faa.gov/ipg>.
9. Ibrahim, Linda, and Curt Wells. Guidelines for Using FAA-iCMM v2.0 and ISO 9001:2000 in Process Improvement. Washington, D.C.: Federal Aviation Administration, 2004 <www.faa.gov/ipg>.
10. Wells, Curt, Linda Ibrahim, and Larry LaBruyere. "A New Approach to Generic Attributes." Systems Engineering 6.4 (2003): 301-308.

Note

1. See <www.sei.cmu.edu> for information on the CMM and CMMI models and their appraisal-related products.
2. See <www.eia.org> or <www.geia.org> for information on Electronics Industries Alliance (EIA) standards, including Systems Engineering Capability Model (EIA 731-1), Systems Engineering Capability Model Appraisal Method (EIA 731-2), and Processes for Engineering a System (ANSI/EIA-632).
3. See <www.iso.ch> for information on standards from ISO and IEC.
4. Ten sources integrated into iCMM v2.0: ISO 9001:2000, EIA/IS 731, Malcolm Baldrige National Quality Award/Presidents Quality Award, CMMI, ISO/IEC TR 15504, ISO/IEC 12207, ISO/IEC CD 15288, and iCMM v1.0, containing SW-CMM, SA-CMM, and SE-CMM.

About the Authors

Linda Ibrahim, Ph.D., is the Federal Aviation Administration's (FAA) chief engineer for Process Improvement where she led development on and is lead author and architect of the FAA-integrated Capability Maturity Model v1.0 and v2.0, and its appraisal method. Ibrahim has been working in software engineering for more than 30 years in the United States, Europe, and the Middle East. She previously worked for the Software Engineering Institute, and is a member of the Capability Maturity Model® Integration Steering Group. Ibrahim has a Bachelor of Arts in mathematics, a Master of Science in information science, and a doctorate degree in electrical engineering.

Federal Aviation Administration
800 Independence AVE SW
Washington, D.C. 20591
Phone: (202) 267-7443
Fax: (202) 267-5069
E-mail: linda.ibrahim@faa.gov

Joan Weszka is the manager of Process and Program Performance at Lockheed Martin's Systems & Software Resource Center, providing consulting and training in areas including process improvement, program management, and engineering. Weszka has more than 25 years of experience in software and systems engineering and program management, and is a member of the Capability Maturity Model® Integration Steering Group. She has a Bachelor of Science in mathematics and a Master of Science in computer science from the University of Maryland.

Lockheed Martin Corporation
Systems & Software
Resource Center
700 North Frederick AVE
Gaithersburg, MD 20879-3328
Phone: (301) 240-7013
Fax: (301) 240-7009
E-mail: joan.weszka@lmco.com

Coming Events

July 13-17
CAV 2004 Computer Aided Verification
Boston, MA
www.dcs.warwick.ac.uk/CAV

July 18-21
8th World Multiconference on Systemics, Cybernetics, and Informatics

Orlando, FL
www.iiisci.org/sci2004

July 21-25
CITSA 2004: Cybernetics and Information Technologies, Systems, and Applications
and
ISAS 2004: the 10th International Conference on Information Systems Analysis and Synthesis
Orlando, FL
www.infocybernetics.org/citsa2004

August 14-17
CCCT Conference: Computing, Communications, and Control Technologies
Austin, TX
www.iiisci.org/ccct2004

August 19-20
2004 ACM-IEEE International Symposium on Empirical Software Engineering
Redondo Beach, CA
www.isese.org

August 23-27
International Conference on Practical Software Quality Techniques
PSQT 2004 North
Minneapolis, MN
www.qualityconferences.com

April 18-21, 2005
2005 Systems and Software Technology Conference

Salt Lake City, UT
www.stc-online.org

Predictable Assembly From Certifiable Components[1]

Scott A. Hissam
Software Engineering Institute

Using predictable assembly from certifiable components is one approach to developing software systems with run-time qualities that are predictable by construction. Predictable assembly combines advances in software component technology and software architecture to automate many engineering activities in constructing predictable component-based systems. In this article, I introduce the concept of predictable assembly and its connection to certifiable components, and provide a brief illustration of early experience with this approach.

Advances in technologies that support software specification and development promise dramatic improvements in the quality of software intensive systems and in the reduced cost of developing (and therefore acquiring) such systems. Progress in two broad areas is particularly noteworthy:

1. **Trusted Software Components.** It is now beyond doubt that a commercial market of software components exists, and will play an increasingly prominent role in the development of Department of Defense (DoD) systems. Recognizing this fact has led to renewed interest in the question of trustworthy components – components that are certified to exhibit known quality standards and to honor their specifications[2, 3].
2. **Analyzable Software Architecture.** While components exhibit various qualities individually, systems having such components exhibit their own emergent qualities. These emergent qualities can only be understood when a system is viewed at a level of abstraction that includes not only components but also, for example, their patterns of interaction. Software architecture technology has emerged as a way for systems designers to address the need for predictable system quality attributes at design time [1].

One theme of predictable assembly from certifiable components is to combine elements of the above two areas to provide an end-to-end method that begins with analyzable design and ends with deployed software systems that satisfy their run-time requirements. An equally important theme is using software component technology as a way of packaging and deploying the capability for predictable assembly from certifiable components into software development houses, and making these technologies easy to use by designers and developers.

This article is written from the vantage of work in predictable assembly from certifiable components (PACC) conducted at the Software Engineering Institute (SEI)[4]. Our goal is to achieve *predictability by construction*, the meaning of which is discussed later in this article. However, our work is best seen as a manifestation of – or perhaps a specialization of – a more fundamental evolution in software development practice, referred to as Model Driven Architecture (MDA) [2][5].

Both PACC and MDA are motivated by the desire to provide an end-to-end flow method from design to deployment. However, our approach to predictable assembly is to restrict designers and developers to a class of designs that are known, by construction, to be analyzable and therefore predictable. The trade off between generality and predictability must of course be made in particular development settings. Where predictability is paramount – for example in real time, secure, or highly available systems – restriction may be warranted.

Predictable Assembly

Assemblies result from composing individual software components into an interconnected collection of parts intended to carry out one or more specific functions. Often, one or more component technologies and/or protocols are used as the common unifying mechanism that permits components to be fitted together. Component technologies may be off-the-shelf such as Microsoft's Component Object Model (COM[6]), Object Management Group's Common Object Request Broker Architecture[7], Sun's Enterprise JavaBeans[8], or home grown. In any case, a common component technology is required to plug two or more components together, but it is not sufficient to ensure that the components will play well together.

To determine whether or not two components will play well together, software engineers typically look at the component's set of inputs, outputs, pre- and post-conditions, and, when available, the description of the component's assumptions about the environment (such as required processor type and speed, available memory, etc). If there is a match, the engineer will fit the two components together and hope everything works, and works well. If they do not match, the engineer may find another component and try again. To verify (or rather, gain confidence) that two components do work well together, the engineer will then test those integrated components to see if they fail. When they do fail, likely it is for reasons other than that which could have been deduced at the time the initial selection was made [3].

Predictable assembly, then, is an approach for integrating individual software components into a collection of parts where critical run-time properties (e.g., performance, safety, etc.) of that collection are reliably predicted. That is, by using predictable assembly it can be known before the actual components are integrated that they will play together with respect to one or more run-time properties of interest. This can be done if the properties of individual software components are known *a priori* to their selection or acquisition. The properties of an individual component can be the following:

- As simple as the execution latency of a function call on the component (when considering the performance of the assembly).
- As complex as a state machine description of the function call itself (when considering the safety of the assembly).

In this approach, it is the properties of individual components that are integrated together rather than the actual components. Therefore it is not necessary to actually acquire a component in advance of making the determination if it will work well together with other components. That determination is aided by a *rea-*

soning framework that is specific to a property of an assembly for which it is desired to predict.

A reasoning framework uses these properties to make a determination if the assembly of those components is well formed with respect to the rules dictated by the reasoning framework. If the assembly is well formed, then the reasoning framework generates a prediction (e.g., see the example in PACC in Action). Further, the prediction can be trusted, as the reasoning framework itself is statistically labeled to generate predictions with a stated accuracy and confidence level.

For software system developers and integrators, predictable assembly means the following:

- Reduced guesswork as to whether or not the component selection made is viable for the context in which the component will be used.
- Assemblies are predictable by construction.
- Greater confidence that components will work well together prior to testing.
- Lower likelihood that redesign, reintegration, and retesting of actual components will be necessary.

The properties that serve as input are specific to the reasoning framework. If the reasoning framework is predicting execution latencies of tasks, then the individual component latencies are required as the properties of input. If the reasoning framework is proving that an assembly is deadlock-free, then the individual component state machine might be the required property of input. From the inputs to the reasoning framework, predictions and acquisition decisions could be made. As such, it is critical that those input properties to the reasoning framework be trusted or, ideally, certified.

Certifiable Components

A component is certifiable if it has properties that can be demonstrated in an objective way. Common examples of this occur in the consumer marketplace. For example, hard disk drive (HDD) manufacturers often provide data sheets that attest to various properties of their products (e.g., seek time, average latency, or mean time between failures). Objectively, an end consumer of one of these HDDs could measure the seek time and average latency of the HDD and know whether or not its manufacturer was telling the truth. Mean time between failures would be harder for the end consumer to independently confirm, as the consumer would need all the historical data from the HDD manufacturer to reproduce the same HDD property. In this example, then, the consumer trusts that the HDD manufacturer has objectively stated these properties, and often treat them as certified properties.

Certification of a component's properties does not necessarily have to come from the component manufacturer. Consider a component that comes from the free/open source software community. An end user would be free to publish a state machine description of that component, and could even publish results that verify the component implementation matches the published state machine. This would make the state machine a certifiable property of that component. Any property of a component that can be demonstrated (in the form of a verifiable proof) or is plausible (in the form of empirical observation) can be the subject of certification.

Certification need not be a pass/fail proposition, although it is frequently treated as such. Descriptive certification of a component property (as opposed to a pass/fail normative certification) is a statement about an objective fact about a component. Revisiting the HDD example, the fact that a particular HDD has an average seek time of < 1 millisecond is not a statement that this HDD is good or bad. It is simply a stated fact, and if the consumer trusts the HDD manufacturer, the consumer can treat it as a certified property. This, then, leaves it to the integrator to determine if the value of the certified property is good enough for the assembly in which the component will be used.

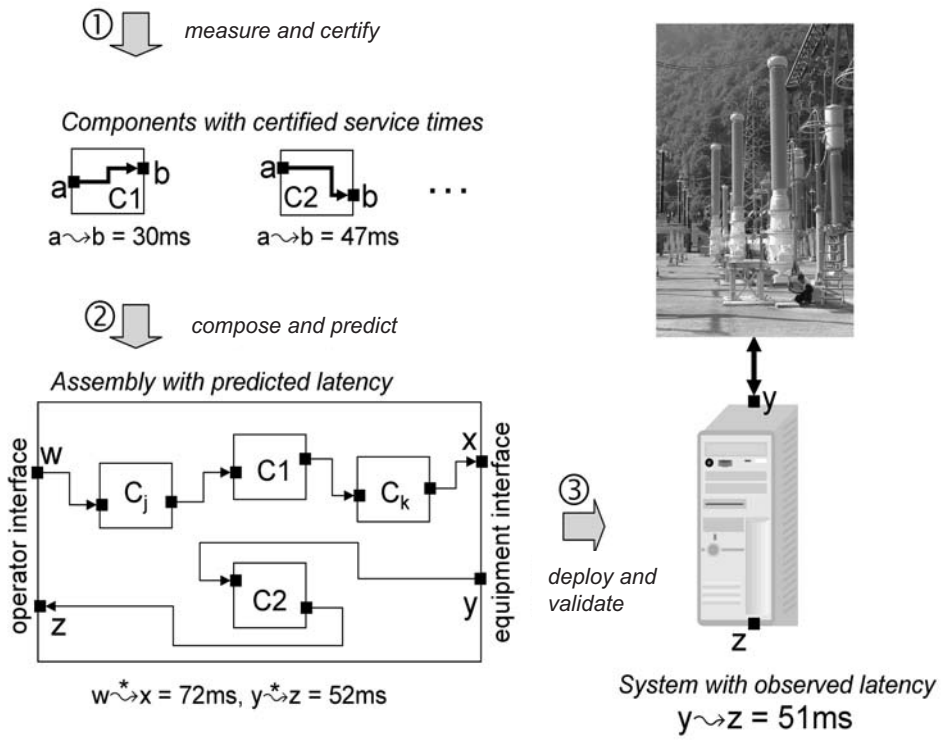
PACC in Action: A Simple Illustration

In this illustration, a software engineer wishes to predict a run-time property, execution latency, of a task with an assembly of components. The illustration is drawn from a proof of feasibility of predictable assembly for power transmission and distribution [4].

A power substation serves several purposes, among which is protection and control of primary equipment such as transformers, circuit breakers, and switches. The task for the software engineer in this illustration is to develop, from software components, a controller for a high-voltage switch. One function of the controller is to provide an interface that allows operators to manually open and close the switch. One activity in this task is to predict the time it takes for a controller to process operator requests, and the time it takes for the controller to report on a change in switch status.

The illustration in Figure 1 presents the gestalt of the software engineering task in terms of predictable assembly. Assume that a set of software components already exists, and that the service time of these components (defined as the time it takes for a component to do its work, assuming no blocking or pre-emp-

Figure 1: *A Predictable Substation Assembly*



tion) has been obtained or certified to a certain degree of trust (① in the figure). The software engineer selects a set of candidate components and composes specifications to produce a model of the controller assembly, which is analyzed and from which the execution latency of a task is predicted (② in the figure).

In the illustration, the connection from *y* to *z* is computed automatically based on the certified latency of C2 in the context of the entire assembly (*w* to *x* in this case) that may introduce blocking and preemption during run time, effecting latency. If the predicted latency satisfies requirements, the components (rather than their specifications) are composed and the resulting assembly is deployed. Predictions are just predictions because there is a possibility that they are wrong, so some validation is required of the deployed assembly (③ in the figure).

This illustration is intended to encapsulate the idea of how predictable assembly can be used in a development setting. What is not shown in Figure 1 is the level of automation supported in the assembly, prediction, and composition processes. In particular, using this example results in the following:

- Latency prediction for user-selected controller operations (e.g., from arrival of an operator request on *w* until the switch is signaled on *x* in Figure 1) is computed automatically from assembly specifications.
- The reasoning framework used to make latency predictions defines precisely what run-time properties of components must be known, and how these properties are specified and obtained. Thus, the properties of components that must be trusted are precisely those that enable predictions of assembly run-time behavior.
- The assumptions underlying the reasoning framework about how components interact with their environment and with each other are made explicit. Assemblies are well formed if they satisfy these assumptions. How well they are formed is checked automatically thus, assembly behavior is predictable by construction.
- The accuracy and reliability of reasoning framework predictions is objectively validated using statistically sound sampling and measurement. The quality of predictions is specified as a confidence interval – e.g., nine out of 10 predictions will have an upper error bound of 3 percent with 95 percent confidence.

Although this illustration is focused on execution latency, our project is concerned with more than just the timing properties of assemblies – e.g., safety and liveness (areas of current work), and reliability and security (areas for future work). Therefore, the technology our project is developing can be applied to many reasoning frameworks.

Status

The initial application of the SEI's PACC approach to predictable assembly was motivated by the challenges of using software component technology in the field of substation automation systems [4]. Although our project developed and validated a prototype infrastructure for predictable assembly, our main objective was exploratory. The primary result was an overall process model for the design, development, and validation for predictable assembly [5].

> *"Predictable assembly from certifiable components is not a radical concept, especially when viewed from the vantage of traditional engineering discipline."*

A secondary result from this work was the development of a measurement and validation infrastructure supporting empirical validation of a reasoning framework – it is the validation of a reasoning framework that quantifies the quality of predictions produced by a reasoning framework for the user. A tertiary result from this work was the development of a prototype for predicting the latency of substation operator commands to a switch controller. This prototype ran on two platforms: a substation operator platform using Microsoft .NET, and a switch controller platform using Microsoft COM. The two platforms communicated through an industrial middleware, Object Linking and Embedding for Process Control[9], and used the International Electrotechnical Commission 61850 Standard for substation automation component type model [6].

Lessons from the initial application included the following:

- Adherence to the invariants demanded by the reasoning framework is vital.
- Development of a reasoning framework is a time-consuming proposition.

The first lesson from this list made it clear that the ability to reason (and ultimately make a prediction) about an assembly of components relies on consistency between what the reasoning framework expects to be true about the assemblies and its constituent components and the assemblies that can be created in the component technology. For example, the reasoning framework expected that components in the assemblies adhere to priority ceiling protocol [7]; however, the human designer did not always adhere to that restriction causing poor predictions. This inconsistency was spotted during validation of the reasoning framework. However, more specific rigor was clearly needed to establish and maintain consistency.

The second application of our approach (in the domain of industrial robot control, which is currently underway) is expanding, technically, to address this lesson with language (Component and Composition Language) and tool (compilers and code generators) support. The key aspect behind this additional suite of tools [8, 9] is to enforce, through automation, consistency between what is built and the invariants required by a reasoning framework.

The second lesson from this list reflects the need for expertise in the mathematical and formal models used as the foundation for any reasoning framework. As our project moves forward, it is broadening its repertoire of reasoning frameworks to include a variety of performance and verification (through model checking) technologies. Our project does this with the end goal to package these reasoning frameworks into a starter-kit to reduce the initial investment needed to create reasoning frameworks, and to make predictable assembly a practical tool for the design and deployment of software with predictable behavior.

Challenges

MDA, or something like it, is inevitable. Our project's specialized approach to MDA focuses on using software component technology to package analyzable architectural design patterns and associated reasoning (analysis) methods. As mentioned earlier, our team is developing methods and tools that will enable the software industry as well as the DoD to introduce predictable assembly from certifiable components into practice. Our team is working to demonstrate the feasibility of this approach in industrial settings, and is

seeking suitable DoD applications for trial use as well.

Although our team believes that it has demonstrated the potential of predictable assembly, there are several challenges that must be met if the ideas are to find widespread use and acceptance:

- Techniques for certifying, and labeling component properties required by reasoning frameworks must be developed.
- The business case for prediction and certification must be established, since the development of an infrastructure for predictable assembly requires upfront investment.
- The engineering methods and technology needed to build and use predictable assembly must be better understood, documented, and supported by commercial tools.

These are serious challenges, but the needs addressed by predictable assembly are real and immediate. Moreover, progress is being made, and not just at the SEI. Academic research[10] [10] and industrial practice [11, 12] are moving in the direction of predictable assembly. Further, guaranteed component quality is increasingly demanded by the marketplace, by societal needs, and by the software community's quest to establish rigorous foundations for software engineering practice.

Summary

Predictable assembly from certifiable components is not a radical concept, especially when viewed from the vantage of traditional engineering discipline. The key principle is to restrict developers to build only systems whose behaviors can be predicted, rather than trying to develop a general-purpose technology that can predict the behavior of any system. Granted, restricting developer freedom has never been an important concern of the software technology marketplace, but with the maturing of the software engineering discipline – and with the self evident importance of software to our safety and standard of living – these market forces may finally be poised to make a change for the better.◆

Acknowledgements

I would like to thank Linda Northrop, Kurt Wallnau, James Ivers, Paulo Merson, Daniel Plakosh, and Jacqueline Hissam for their helpful reviews.

References

1. Bass, L., P. Clements, and R. Kazman. Software Architecture in Practice. 2nd ed. Reading, MA: Addison-Wesley, 2003.
2. Mellor, S., and M. Balcer. Executable UML: A Foundation for Model Driven Architecture. Reading, MA: Addison-Wesley, 2002.
3. Hissam, S., and D. Carney. "Isolating Faults in Complex COTS-Based Systems." Journal of Software Maintenance: Research and Practice. John Wiley & Sons, Ltd., Mar. 1999. 183-199.
4. Hissam, S., et al. "Predictable Assembly of Substation Automation Systems: An Experiment Report." CMU/SEI-2002-TR-031. Pittsburgh, PA: Software Engineering Institute, 2002. <www.sei.cmu.edu/publications/documents/02.reports/02tr031.html>.
5. Wallnau, K. "Volume III: A Technology for Predictable Assembly From Certifiable Components." CMU/SEI-2003-TR-009. Pittsburgh, PA: Software Engineering Institute, 2003 <www.sei.cmu.edu/publications/documents/03.reports/03tr009.html>.
6. International Electrotechnical Commission. "Communications Networks and Systems in Substations." Working Draft for International Standard IEC 61850-1.10. Geneva, Switzerland: International Electrotechnical Commission, 2002.
7. Goodenough, J., and L. Sha. "The Priority Ceiling Protocol: A Method for Minimizing the Blocking of High-Priority Ada Tasks." CMU/SEI-88-SR-004. Pittsburgh, PA. Software Engineering Institute, 1988. <www.sei.cmu.edu/publications/documents/88.reports/88.sr.004. html>.
8. Wallnau, K., and J. Ivers. "Snapshot of CCL: A Language for Predictable Assembly." CMU/SEI-2003-TN-025. Pittsburgh, PA: Software Engineering Institute, 2003 <www.sei.cmu.edu/publications/documents/03.reports/03tn025.html>.
9. Hissam, S., and J. Ivers. "PECT Infrastructure: A Rough Sketch." CMU/SEI-2002-TN-033. Pittsburgh, PA: Software Engineering Institute, 2002 <www.sei.cmu.edu/publications/documents/02.reports/02tn033.html>.
10. Meyer, B. The Grand Challenge of Trusted Components. Proc. of 25th International Conference on Software Engineering, Portland, OR, May 2003. New York: IEEE Computer Press, 2003.
11. Soley, R., et al. "Model Driven Architecture." White Paper Draft 3.2. Needham, MA: Object Management Group, 27 Nov. 2000.
12. Microsoft. "Foundations of Software Engineering." Redmond, WA: Microsoft Research, 2003 <http://research.microsoft.com/fse>.

Notes

1. Sponsored by the U.S. Department of Defense.
2. See <http://niap.nist.gov/cc-scheme> for a U.S. government-sponsored effort to establish certification criteria for security-related aspects of components.
3. For Bertrand Meyer's, et al. take on the issue of trusted components, see <http://archive.eiffel.com/doc/manuals/technology/bmarticles/computer/trusted/page.html>.
4. See <www.sei.cmu.edu/pacc> for details.
5. See <www.omg.org/mda> for details.
6. See <www.microsoft.com/com> for details.
7. See <www.omg.org/gettingstarted> for details.
8. See <http://java.sun.com/products/ejb> for details.
9. See <www.opcfoundation.org/01_ about/01_whatis.asp> for details.
10. Department of Computer Science – Research. Swiss Federal Institute of Technology, Zurich, Switzerland <www.inf.ethz.ch/research/institutes/group.php?grp=Meyer#Anchor-Trusted>.

About the Author

Scott A. Hissam is a senior member of the technical staff for the Software Engineering Institute at Carnegie Mellon University. Hissam conducts research on component-based software engineering and open source software. He is also an adjunct faculty member of the University of Pittsburgh. Previously, he held positions at Lockheed Martin, Bell Atlantic, and the U.S. Department of Defense. Hissam is co-author of "Building Systems from Commercial Components" and has been published in international journals, including *IEEE Internet Computing* and *Journal of Software Maintenance*. He has a Bachelor of Science in computer science from West Virginia University.

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890
Phone: (412) 268-6526
Fax: (412) 268-5758
E-mail: shissam@sei.cmu.edu

Software Engineering for End-User Programmers

Dr. Curtis Cook, Shreenivasarao Prabhakararao, Martin Main, Mike Durham, Dr. Margaret Burnett, and Dr. Gregg Rothermel
Oregon State University

It is estimated that by 2005, there will be 55 million end-user programmers compared to 2.75 million professional programmers. Even though end-user programs have the same reliability problems, software engineering research has largely ignored the end-user community. Because end users are different from professional programmers in motivation, background, and interests, the end-user community cannot be served by repacking tools and techniques developed for professional programmers. This article describes our work in developing software engineering devices for spreadsheet developers, one of the largest classes of end-user programmers.

Software engineering research has focused on aiding programmers throughout the software development and maintenance process. However, this focus has been on professional programmers and has largely ignored the sizeable end-user programmer community. It is predicted that by 2005 in the United States alone there will be 55 million end-user programmers compared to 2.75 million professional programmers [1]. The programming systems used by these end users include spreadsheets, web authoring tools, scientific visualization languages, and graphical languages for creating educational simulations.

It should not be surprising that a high percentage of end-user programs contain errors that can have significant economic impact. For example, a Texas oil and gas company lost millions of dollars in an acquisition deal because of spreadsheet errors [2]. In error data collected from field audit reports of real-world spreadsheets, Panko [2] reported that 20 percent to 40 percent of the spreadsheets contained errors, and errors were as high as 90 percent in some of the financial models reviewed. In empirical studies involving both experienced and inexperienced spreadsheet developers, he found that over 60 percent of the spreadsheets created by the participants contained errors. Compounding the reliability problem is the unwarranted confidence of end users that their spreadsheets do not contain errors [3].

What is surprising is that software engineering research has paid little attention to spreadsheet programmers and other end-user programmers. Our research has focused on the spreadsheet paradigm, the most widely used and studied end-user programming paradigm. Our intent is to bring some of the advances in software engineering research to these end users without requiring that they first learn the underlying software engineering theory and principles. We call this concept *end-user software engineering*.

In this article, we first point out some of the unique characteristics of spreadsheet end users. This serves two purposes. First, it shows that traditional software engineering techniques must be modified for end users; second, it provides a context for understanding the methodologies and tools we have developed as part of end-user software engineering. These include the *What You See Is What You Test* (WYSIWYT) methodology that provides visual feedback to end users about how much of their spreadsheets have been tested (e.g., degree of testing of their spreadsheets), a *Help Me Test* device that automatically generates test cases, and finally an approach for supporting assertions in end-user software. We present the devices and briefly describe a series of empirical studies that validate our efforts and conclude with a suggested follow-up.

End-User Characteristics

The most obvious difference between professional programmers and end-user programmers is programming experience and background. A high percentage of spreadsheet programmers have little or no programming experience. They view a spreadsheet as a tool to help them solve their problems and regard computers "as a means to an end rather than objects of intrinsic interest" [4].

Hence in adapting a software engineering technique for spreadsheet end users, it is unreasonable to expect them to have the time or interest to learn the underlying theory. Spreadsheet end users are accustomed to working in an incremental fashion in a highly interactive and visual environment with immediate feedback. Further, spreadsheets are usually created in an ad-hoc manner without a clear design plan or formal specification [5]. Even though the spreadsheet creator has a mental model of how it should work, most often it is not explicitly specified, and the actual spreadsheet is only an approximation of the model. Thus any technique developed should require a minimum of training, not assume a programming background or formal problem specifications, and be compatible with the incremental working style.

What We Have Done

Our work has been guided by the above end-user characteristics. We have prototyped our methodology and tools in the spreadsheet research language Forms/3 [6] because we have access to the implementation of Forms/3, and thus we can implement and experiment within that environment. Further, by working with Forms/3 we can investigate not only language features common in commercial spreadsheet languages but also advanced language features found in research spreadsheet languages.

In Forms/3, as in other spreadsheet languages, spreadsheets are a collection of cells and each cell's value is defined by the cell's formula. A programmer receives immediate feedback about a cell's value after the cell formula is entered. Figure 1 shows a Forms/3 spreadsheet that computes student grades based on quiz and extra credit scores. Three differences between Forms/3 and commercial spreadsheets such as Excel are that cells can have meaningful names, more than one cell formula can be displayed at a time, and the cells do not have to be laid out in a grid and can be positioned anywhere on the screen. None of these differences are required for or affect the end-user software engineering devices presented here.

The WYSIWYT Methodology

The WYSIWYT [7] methodology gives end users visual feedback about the degree of testing of individual cells and the entire spreadsheet. The WYSIWYT methodology is based on definition-use associations (du-associations) in a spreadsheet that link a defining expression in a

cell formula (definition) with expressions in other cell formulas that reference (use) the defined cell. See [7] for more details.

The WYSIWYT methodology provides visual feedback about the extent to which du-associations have been covered by tests by means of cell border colors. A percent-tested indicator at the upper right of the spreadsheet gives the percent of du-associations that have been covered. A red cell border (*Total_Score*, *LetterGrade, ErrorsExist?*) means none of the du-associations for the cell have been covered. A blue border (*avg*) means all of the du-associations have been covered, and shades of purple (*EC_Award*) mean some of the du-associations have been covered. Via tool tips, the end users can learn that a red cell border means that a cell is untested, blue means fully tested, and shades of purple mean partially tested.

An end user can also display arrows that indicate dependencies (du-associations) between cells and cell formulas. The arrows follow the same color scheme as cell borders. The arrows reveal the degree of testing at the du-association level, but they are optional; users do not have to think about testing at the du-association level unless they prefer it. Arrows for cell *ErrorsExist?* displayed in Figure 1 indicate a partial degree of testing. Since the formula for this cell is displayed, the arrows point to the cell references in the formula and from the formula to uses of the cell.

The WYSIWYT visual devices keep the user continually informed about the degree of testing of the spreadsheet, draw attention to untested parts of the evolving spreadsheet, and suggest where testing will cover new situations. As cell formulas are modified or new cells added, du-associations are added, deleted, or modified; these changes to du-associations are immediately reflected in the cell border and arrow colors and the percent-tested indicator (upper right indicator).

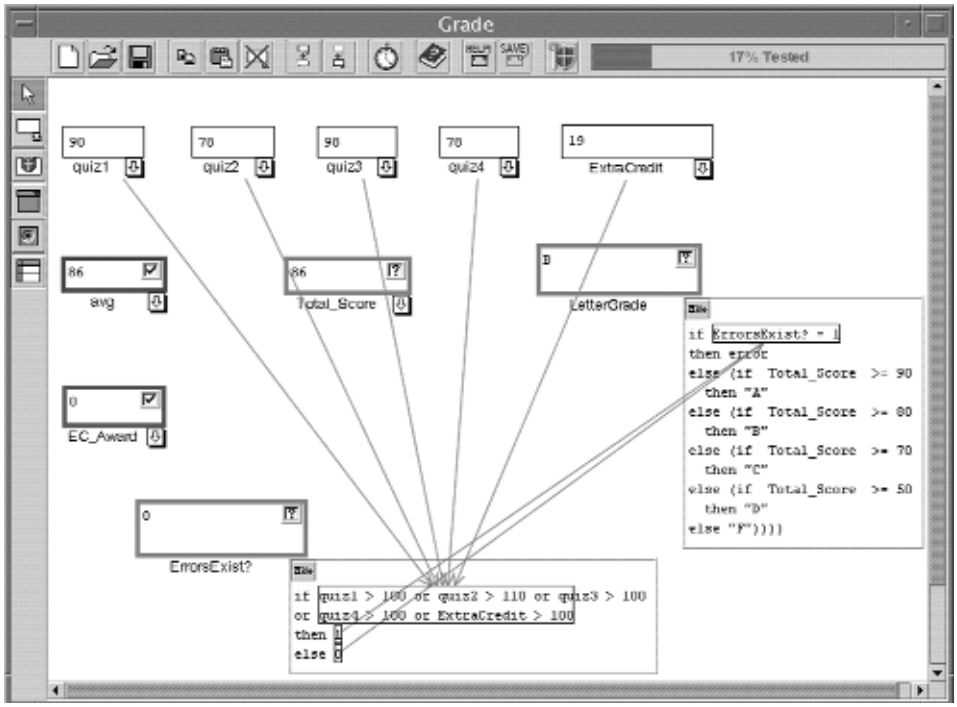


Figure 1: *A Forms/3 Grades Spreadsheet*

Help Me Test

As described to this point, the WYSIWYT relies solely on the skill of the end user to develop test cases for his or her spreadsheets. Sometimes the end user will know from the WYSIWYT feedback that a spreadsheet is not fully tested, but will be unable to find a set of inputs for a new situation. To aid end users in finding appropriate input values for these situations, we have integrated a Help Me Test device that the user can invoke to find a test case. When Help Me Test succeeds, it stops and highlights the input cells that have been changed and the cells that now cover new situations. Figure 2 shows only the output in the Help Me Test window when invoked for cell *EC_Award* in the Grades spreadsheet and not the cells in the spreadsheet that have been changed. The user can then make testing decisions about some or all of these cells. A user can invoke Help Me Test for the entire spreadsheet, a single cell, or a particular arrow.

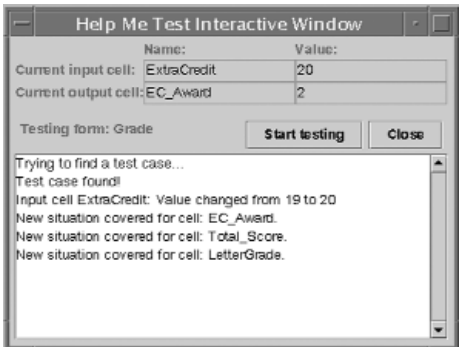
Assertions

Assertions – statements about the properties of a program – are used by professional programmers to prove their programs are correct and to help detect errors. When creating a spreadsheet, the user has a mental model of properties it should have and how it should operate. One approximation of this model is the formulas they enter, but unfortunately these formulas may contain inconsistencies or faults. These formulas, however, are only one representation of the user's model of the problem and its solution: They contain information on how to generate the desired result, but do not provide ways for the user to communicate other properties. Traditionally, assertions in the form of preconditions, post conditions, and invariants have fulfilled this need for professional programmers, providing a method of making explicit the properties the programmers expect of their program logic, providing a reason about integrity of their logic and providing a way to catch exceptions.

While these forms of assertions may aid professional programmers, their syntax and Boolean expressions are inappropriate for most end users. Our approach attempts to provide the same advantages to end-user programmers, but is different from traditional approaches in that ours is a component of our integrated set of software engineering features specifically designed for end users. As part of the incremental end-user spreadsheet development, the user can enter a few assertions and see the effects. Our assertions look like simple ranges, but because they include open and closed ranges, *and*, *or*, and references to cells, this syntax allows a fairly powerful set of assertion types [8].

There are two types of assertions: user-entered and system-generated. User-entered assertions are those explicitly entered by the user while the generated assertions result from propagating assertions through formulas in the direction of dataflow using logic and interval arithmetic. User-entered and system-generated assertions are stacked on the top of the cells in Figure 3 (see Page 22). The

Figure 2: *Help Me Test Window*



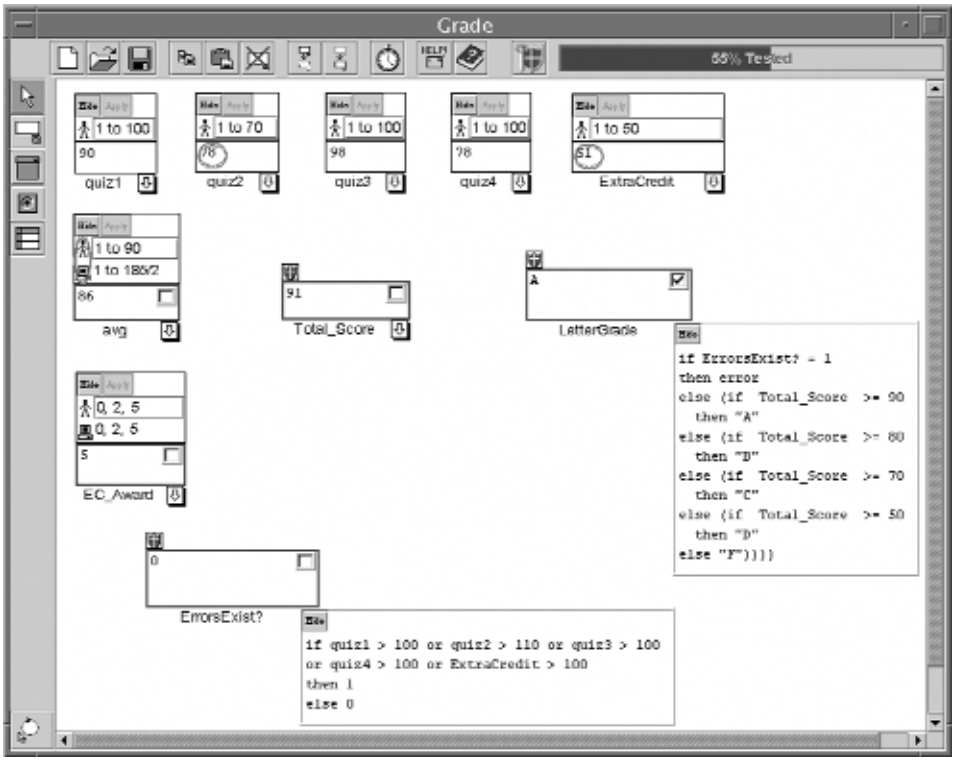


Figure 3: *User-Entered and System-Generated Assertions in Grades Spreadsheet*

top row of cells is simply input cells with constant values as their formulas. Cell *ExtraCredit* has a user-entered assertion (stick figure icon) from one to 50 while the user-entered and system-generated (computer icon) assertions for cell *EC_Award* are the three integer values zero, two, or five. Assertions help users detect errors through assertion conflicts (user and system assertions disagree) and value violations (cell value outside of range). To draw the user's attention to possible errors, red ovals circle assertion conflicts and value violations. Cells *quiz2* and *ExtraCredit* have value violations and the cell *avg* has an assertion conflict in Figure 3.

To introduce users to the idea of entering assertions, Help Me Test provides suggested assertions on some cells that do not yet have them. When users run Help Me Test to get new test inputs, our empirical work showed that these suggested assertions were effective in inducing them to use assertions while debugging [9].

Commercial spreadsheets such as Microsoft Excel have a data validation feature that bears a surface similarity to assertions in our environment. However, these commercial spreadsheets do not propagate assertions, do not automatically display assertions, and do not update the display of assertion violations when changes are made. In short, their assertions are data entry checks, whereas ours form an ever-present reasoning mechanism that watches over all the cells at all times.

Validation

We have used empirical studies both to demonstrate that our methodology and tools do indeed aid end users in testing, debugging, and maintaining their spreadsheets and to gain a better understanding of how end users work and how our devices help them. In nearly all of these studies we have used sophomore and junior business majors as subjects.

Two controlled experiments [10, 11] showed that subjects using the WYSIWYT methodology tested significantly better (higher coverage, fewer redundant tests) and were significantly more successful in a maintenance task (more correct modifications, more testing) than subjects without the WYSIWYT methodology. In a debugging study [12], we found that WYSIWYT subjects using assertions found significantly more bugs and found them faster than WYSIWYT subjects without assertions. A follow-up study [9] showed that end users elected to enter assertions of the type described in this article, and did so quite accurately.

We have also conducted several think-aloud studies during which we observe subject behavior and record subject verbalizations as they perform the experimental task. These studies provide insight into their thought processes and strategies. Our think-aloud studies have found that end-users understood assertions and could effectively use them in a maintenance task [8], and that end-users with WYSIWYT and Help Me Test were more effective and more efficient in a modification task than end-users with only WYSIWYT [8]. In all of our experiments, the subjects using our end-user software engineering devices showed a more appropriate level of confidence about whether their spreadsheets contained errors.

Conclusions

Software engineering research has largely ignored the end-user community in spite of the fact that there will soon be 20 times as many end-user programmers as professional programmers. Yet, it should not be a surprise that end-user programs have the same correctness problems. Because end users are different from professional programmers in background, motivation, and interests, the end-user community cannot be served by simply repackaging techniques and tools developed for professional programmers. Instead, the methodologies, tools, and techniques developed for end users must take these differences into account.

In this article we have described our approach in developing software engineering devices for spreadsheet users, which has been met with considerable success. We advocate that spreadsheet languages contain some of the devices we have developed, and we believe our approach holds promise for those developing tools and techniques for other types of end-user software. We welcome the opportunity to collaborate with others interested in this work. If you are interested in either theoretical or practical follow-up, please contact author Dr. Curtis Cook.◆

References

1. Boehm, B., E. Horowitz, R. Madachy, D. Riefer, B. Clark, B. Steece, A.W. Brown, S. Chulani, and C. Abts. Software Cost Estimation With COCOMO II. Englewood Cliffs, N.J.: Prentice-Hall, 2000.
2. Panko, R. "What We Know About Spreadsheet Errors." Journal of End User Computing Spring 1998: 15-21.
3. Brown, P., and J. Gould. "Experimental Study of People Creating Spreadsheets." ACM Transactions on Office Information Systems 5.3 (July 1987): 258-272.
4. Nardi, B., and J. Miller. "Twinkling Lights and Nested Loops: Distributed Problem Solving and Spreadsheet Development." Int. J. Man-Machine

Studies 34 (1991): 161-184.
5. Ronen, B., R. Palley, and H. Lucas. "Spreadsheet Analysis and Design." Communications of the ACM 32.1 (Jan. 1989): 84-93.
6. Burnett, M., J. Atwood, R. Djang, H. Gottfried, J. Reichwein, and S. Yang. "Forms/3: A First-Order Visual Language to Explore the Boundaries of the Spreadsheet Paradigm." Journal of Functional Programming 11.2 (Mar. 2001): 155-206.
7. Rothermel, G., L. Li, C. DuPuis, and M. Burnett. What You See Is What You Test: A Methodology for Testing Form-Based Visual Programs. Proc. of 20th International Conference on Software Engineering. Kyoto, Japan. Apr. 1998: 198-207 <http://cs.oregon state.edu/~burnett/ITR2000>.
8. Rothermel, K., C. Cook, M. Burnett, J. Schonfeld, T.R.G. Green, and G. Rothermel. WYSIWYT Testing in the Spreadsheet Paradigm: An Empirical Evaluation. Proc. of the 22nd International Conference on Software Engineering. Kyoto, Japan. June 2000: 230-239.
9. Wallace, C., C. Cook, J. Summet, and M. Burnett. Assertions in End-User Software Engineering: A Think-Aloud Study. Proc. of IEEE 2002 Symposia on Human Centric Computing Languages and Environments (HCC 2002). Arlington, VA, Sept. 3-6, 2002: 63-65.
10. Wilson, A., M. Burnett, L. Beckwith, O. Granatir, L. Casburn, C. Cook, M. Durham, and G. Rothermel. Harnessing Curiosity to Increase Correctness in End-User Programming. Proc. of ACM Conference on Human Factors in Computing Systems. Ft. Lauderdale, FL, Apr. 5-10, 2003.
11. Krishna, V., C. Cook, D. Keller, J. Cantrell, C. Wallace, M. Burnett, and G. Rothermel. Incorporating Incremental Validations and Impact Analysis Into Spreadsheet Maintenance: An Empirical Study. Proc. 25th IEEE International Conference on Software Maintenance. Florence, Italy, Nov. 2001: 72-78.
12. Burnett, M., C. Cook, O. Pendse, G. Rothermel, J. Summet, and C. Wallace. End-User Software Engineering With Assertions in the Spreadsheet Paradigm. Proc. of the International Conference on Software Engineering. Portland, OR, May 2003.

About the Authors

Curtis Cook, Ph.D., is professor of computer science at Oregon State University. He has more than 20 years of research and experience in software complexity metrics, program understanding, and software quality. Cook is a member of the editorial board of the *Software Quality Journal*. He has a doctorate in computer science from the University of Iowa.

Computer Science Department
Oregon State University
Corvallis, OR 97331-3202
Phone: (541) 737-5564
Fax: (541) 737-3014
E-mail: cook@cs.orst.edu

Martin Main is a senior and undergraduate research assistant in Computer Science at Oregon State University. Prior to returning to college studies, Martin was a recording engineer assisting on major label recordings. The current shift in that industry into computer usage sparked Main's interest in how computers are used by society as a whole.

E-mail: mainma@cs.orst.edu

Margaret Burnett, Ph.D., is an associate professor in the Computer Science Department at Oregon State University. She previously worked for several years in industry. Her research interests are where programming languages, human-computer interaction, and software engineering meet, namely in visual programming languages and in how programming language and software engineering research can be applied to support end-user programming. Burnett received the National Science Foundation's Young Investigator Award for her work in visual programming languages. She has a doctorate in computer science from the University of Kansas.

E-mail: burnett@cs.orst.edu

Shreenivasarao Prabhakararao is a graduate student in the Computer Science Department at Oregon State University. He works as a research assistant in the Forms/3 research group. His research interests are software engineering, software testing, and empirical studies. Prabhakararao has a bachelor's degree from Osmania University, India, and a master's degree in computer applications from Birla Institute of Technology, Mesra, India.

E-mail: prabhash@cs.orst.edu

Mike Durham is a senior and undergraduate research assistant with Oregon State University's Computer Science department. He contributed to a recently published paper on the curiosity and behavior of end users. Durham's research interests include human-computer interaction, end-user software engineering, and psychology.

E-mail: durhammi@cs.orst.edu

Gregg Rothermel, Ph.D., is an associate professor in the Computer Science Department at Oregon State University. His research interests include software engineering and program analysis, with an emphasis on software maintenance and testing. Rothermel is a recipient of the National Science Foundation's Faculty Early Career Development Award and is associate editor for *IEEE Transactions on Software Engineering*. He has a doctorate in computer science from Clemson University.

E-mail: grother@cs.orst.edu

Competitiveness Versus Security

Don O'Neill
Center for National Software Studies

Cybersecurity threats and vulnerabilities are increasing in number and sophistication, but security readiness is hampered by vendors neglectful in product trustworthiness and by an inadequate user commitment to security readiness. Unwise legislation, inadequate public-private collaboration, a patchwork of government regulatory infrastructure, and lack of business incentive completing this inhospitable environment matches shortfalls in technical architecture, product trustworthiness, and security best practices. The debate over who pays for cybersecurity is tilted toward industry and its responsibility to remain competitive, exposing the nation's critical software infrastructure to predictable security threats. Vendors must make the sacrifices needed to eliminate vulnerabilities; users must invest in resistance, recognition, and reconstitution; and government must communicate, legislate, and regulate to rebalance the business calculation toward security. The community must forge a shared vision spanning realistic assumptions about threats and vulnerabilities and the policy steps needed to achieve survivability.

We are experiencing the fallout from the lunge toward a paperless society without a technology infrastructure. As McNamara said during the Vietnam War, "If you don't watch the periphery, it will soon become the center" [1]. Security has become the center but a center that spans many dimensions.

Cybersecurity has many dimensions, and currently players are free to choose the dimension that best suits their background, experience, interest, or business objective. The challenge facing the country is to frame the issue realistically, to distill those factors that impact on the national interest, and to do so with intellectual honesty and no self-interest. In large measure, we are engaged in operation barn door, and the horse has already left.

What are the dimensions of security?

- It spans threats, vulnerabilities, and readiness.
- It spans the industry's underlying software architecture and environment, and its inability to field trustworthy software systems.
- It spans industry best practices and certification of processes, people, and products.
- It spans the private and public sector and the tensions between them.
- It spans legislative directions with its unintended consequences that impact security.
- It spans the government regulatory infrastructure.
- It spans business with its lack of an essential driving incentive to promote security.

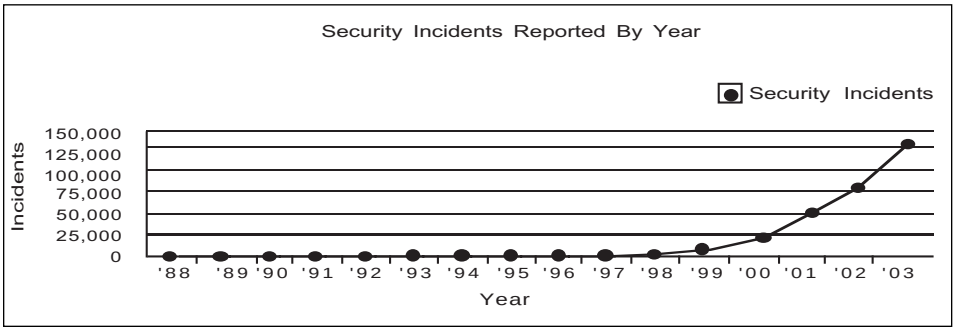
The following sections discuss these dimensions of security in more detail.

Threats, Vulnerabilities, and Readiness

Security spans threats, vulnerabilities, and readiness. The primary software security focus needs to shift from threats and vulnerability to readiness and survivability. Threats are not well understood. Even as we struggle to determine the profile of future incidents, the analysis of past incidents yields only an incomplete and sometimes contradictory profile [1].

The number of security incidents reported to the Computer Emergency Response Team (CERT) Coordination Center has doubled in recent years (see Figure 1). In 2003, 137,529 incidents were reported compared to 82,092 in 2002. In 2001, 52,659 incidents were reported compared to 21,756 in 2000 and 9,859 in 1999. Cyberattack tools permit sophisticated attacks to be carried out by unsophisticated intruders with minimum knowledge who are supported by 30,000 hacker sites on the Web with help and downloadable scripts.

Figure 1: *Security Incidents Reported*



Even as we struggle to determine the profile of future incidents, the analysis of past incidents yields only an incomplete and sometimes contradictory profile. Ninety percent of security threats exploit known flaws, 60 percent are random, and 40 percent are targeted, but the degree of persistence is unknown. While probably 100 percent of U.S. enterprises are attacked, only 30 percent admit to being attacked, perhaps because insiders carry out 70 percent of these attacks. Interestingly, 17 percent of attacks are attributed to industrial espionage and competitive intelligence. What security threats have you experienced?

It is the industry's software products that make us vulnerable to cyberattack [2]. Current vulnerabilities are predominately in implementation not design. These are examples of neglect and stem from unanticipated input, incorrect usage of protocols and connectivity, and accepting vendor default settings. Understanding these vulnerabilities involves chasing down execution paths and their uncountable large number of possibilities.

Vulnerabilities abound. There were 5,000 vulnerabilities identified in 2001, and aaproximately 4,000 vulnerabilities in 2002 alone. The same 30,000 hacker Web sites support these vulnerabilities. Also, industry dependence on Microsoft products with its large pool of users and its common and numerous vulnerabilities greatly facilitates security intrusion into the nation's critical infrastructure, accounting for 90 percent of all vulnerabilities [3].

When it comes to trustworthy software products, Microsoft has forfeited the right to look us in the face. What vulnerabilities are you aware of in actual practice?

Future vulnerabilities may find their way into designs. Security markup language initiatives for common authentication and authorization and those capable of selective word protection in text are innovative. However, these efforts are not validated and there is a lack of research directed at the end-to-end validation of Internet services. Can data streaming through the Internet be tampered with en route, resulting in a security exposure now or later?

If you discover a new vulnerability, what should you do? The reporting of vulnerabilities is in disarray. One vendor has threatened to sue researchers who publicize its security vulnerabilities. The Critical Infrastructure Protection Board advises researchers to contact the vendor before vulnerability is discussed publicly. If the vendor does not respond, the second place to contact is the CERT Coordination Center. Finally, the third place is the Critical Infrastructure Protection Board itself. Clearly a single, independent office should receive all reports and be accountable for analysis, disposition, status, and dissemination of vulnerabilities. These people have invented a strange concept of responsibility.

Regarding readiness, security must be designed in; it cannot be bolted on. Beyond that, there is little consensus on what it means to be ready. Some of the industry approaches to readiness are simply wrong. Some say that security depends on the people doing the protecting, but security cannot be outsourced. Some say security is a journey, not a destination. This brings to mind the saying, "If you don't have a map, any road will do." Is it the destination that is unknown or the road to reach it? Many are treating security as a process improvement activity. After 15 years, industry software process improvement has succeeded in stranding 68 percent of its U.S. practitioners at maturity Levels 1 and 2, below the threshold of competent software engineering, which is Level 3 [4]. Others view security as a risk-management exercise. Hello! We need to be secure now if we are to avoid the digital Pearl Harbor predicted by government officials.

Architecture and Trustworthiness

Security spans the industry's underlying software architecture [5, 6] and environment and its inability to field trustworthy software systems [7]. Industry must make the technical sacrifices needed to achieve enterprise security. Security may require sacrificing certain preferred attributes of trustworthy software systems. For example, openness, interoperability, and modifiability facilitate security intrusions.

In addition, security may require sacrificing certain architectural styles in favor of those that facilitate ease of deterministic recovery and reconstitution following a security intrusion. How many are considering moving from fat clients to thin clients? What technical sacrifices have you made?

Best Practices and Certification

Security spans industry best practices and certification of processes, people, and products. The primary software security focus on industry practices and certification must shift from process and people to product. Industry software configuration management practice is poor, and patches are made without adequate testing. Beyond that, the industry practice is to procrastinate on implementing security patches because upgrades lead to problems, and personnel to test and retest are in short supply. What has been your experience? What is the typical frequency of release for your system upgrades?

> *"The primary software security focus needs to shift from threats and vulnerability to readiness and survivability. Threats are not well understood."*

Private and Public Sector

Security spans the private and public sector and the tensions between them. It is necessary to trade knowledge for power in seeking common ground in the public-private collaboration. There is a public and private consensus that industry must take the lead in addressing security. If the private sector does not come up with market-driven security standards, then government will step up its regulatory pace. However, the government itself has earned failing grades on security readiness [8]. In addition, the private sector is reluctant to report security intrusions to the government due to the Freedom of Information Act. Has your enterprise reported any security incidents?

Legislative Directions

Security spans legislative directions with their unintended consequences that impact security. It is necessary to revise the legislative actions whose consequences are impacting national security. Unintended consequences have accompanied the Uniform Computer Information Transaction Act, the H1B High Tech Immigration Visa Program, the Clinger-Cohen Act, and the Freedom of Information Act.

The availability of security liability insurance might diminish the incentive to improve the software security infrastructure. Currently insurers lack actuarial data on software security, and may demand compliance with good security practice as a prerequisite to underwriting insurance. Software companies often operate as services and are not subject to product liability. Nevertheless, contractors may be reluctant to support government security initiatives without indemnification from third party liability. Are these topics being discussed in your organization?

Government Regulatory Infrastructure

Security spans the government regulatory infrastructure. An enterprise must consider the security cost and information disclosure risk in working with the government. National Security Telecommunications and Information Systems Security Policy No. 11 requires that all commercial off-the-shelf products must be certified by one of several agencies. These are software products that process, store, display, or transmit national security information. It became effective in July 2002.

Presidential Decision Directive (PDD) 63 is intended to promote cooperation between industry and government. The interconnection of the various sectors of the nation's critical infrastructure introduces the risk of cascading consequences following a terrorist attack whether a physical attack or cyberattack. To counter this threat, Information Sharing and Analysis Centers have been created to gather, analyze, and disseminate information and promote public-private cooperation. However, the Freedom of Information Act is throttling the willingness of industry to participate fully and share openly.

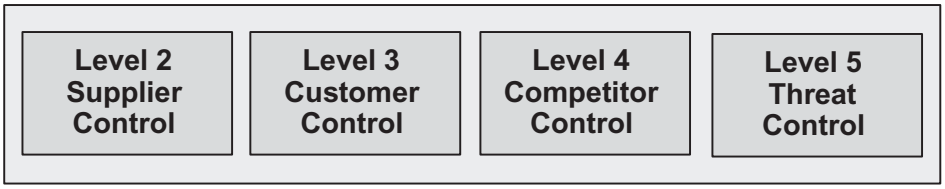


Figure 2: *Levels of Global Software Competitiveness*

Compliance with PDD 63 is achieved through vulnerability assessments using the Information Security Assessment Training and Rating System administered by several organizations.

The Government Information Security Reform Act requires government agencies to integrate security programs into their computer network and capital investment plans. While the price of noncompliance is a budget cut, heads of government agencies lack the skilled staff to comply.

Business Incentive

Security spans business with its lack of an essential driving incentive to promote security. It is necessary to provide effective mechanisms that tilt the essential business calculation from cost effectiveness and competitiveness to trustworthiness, survivability, and security. Enterprise management is driven by *quicker*, *better*, *cheaper* and cost-effective software practices that enhance competitiveness while increasing security risk. Even quality concerns register with enterprise management 10 times higher than security concerns. The high cost of security readiness and the perceived low probability of impact due to security intrusion conspire to promote inaction despite that $13 billion in impact was attributed to security intrusion in 2001. The enterprise must analyze what is to be protected and how important it is to be protected. What needs to be protected in your organization?

The scope of topics under the security tent is broad and deep; consequently, there are no experts. Organizations are now assigning chief security officers to address security in an effort to fence off the blame for this high-risk area. Stovepipe knowledge is increasing with respect to past and current threats and vulnerabilities, but understanding and practicing readiness are lagging. Security threats come from unexpected places. This makes risk management difficult.

The attempt to get a balanced security risk-management program leads to nuanced approaches that look good under the uncritical light of management review but buckle under the intense glare of the factory floor and operating center. A collection of 90 percent approaches does not yield a 100 percent solution. When there is order, incremental change and process improvement can succeed; but when things are in disarray, the practice of tilting borderline practices towards a center line proves inadequate. The antidote for security threats is survivability. For enterprises with software operations at the center of the nation's critical infrastructure, nothing else will do.

Levels of Competitiveness

The government is responsible for prosperity, and industry is responsible for competitiveness. The leading indicators of prosperity span competitiveness, security, and infrastructure because without security and infrastructure, competitiveness cannot be achieved [9]. The Council on Competitiveness in Washington, D.C., defines competitiveness "as the capacity of a nation's goods and services to meet the test of international markets while maintaining or boosting the real income of its citizens" [10].

In software, competitiveness is achieved by providing fuel, setting direction, and controlling the environment, including personnel resources, customer satisfaction and added value, competitors and new entrants, and event threats and change [11]. There are five levels of global software competitiveness (see Figure 2):

- Level 1 is the absence of expectation, achievement, and engagement in the conversation on global software competitiveness.
- Level 2 is the availability of personnel skills and resources and their deployment.
- Level 3 is value to the customer derived through vigorous competition for current market niche with mature products that deliver value and earn customer satisfaction.
- Level 4 is competing for the future by setting the industry standard and practicing reuse and domain architecture technology to meet it.
- Level 5 is managing change and controlling event threats through strategic software management that raises the ability to improve to a core competence.

Who Pays the Bill?

The government has bought in on the security problem, but industry has not yet been sold. Industry appears to treat security as either a business challenge or a business opportunity, but it has not made a commitment to the essential investment of infrastructure. There is a public and private consensus that industry must lead in addressing security; however, with industry slow to take the lead, the government can be heard rattling its regulatory sword in the form of standards.

There is an important national debate on cybersecurity. It centers on who pays the bill, the private or public sector. On one hand, the public sector argues that security and competitiveness move together, therefore, the private sector should pay the cost to be competitive. On the other hand, the private sector argues that security costs too much, and the probability of occurrence is too low to force the investment especially during the period of economic recovery.

The Trade-Off Factors

As Deming[1] taught us, there is no substitute for superior knowledge. The knowledge required in this trade-off revolves around the practices and factors that enhance both competitiveness and security and those that enhance one at the expense of the other (see Table 1).

Three types of practices and factors are used to frame the issue, including trustworthiness, cost effectiveness, and survivability. Trustworthiness revolves around an engineering practice that tolerates change and yields dependability of results [7]. Well-engineered software products are complete, correct, consistent, conforming, traceable, simple not complex, scalable, predictable, and usable.

Table 1: *Trade-Off Factors*

	Competitiveness	Security
Engineering Practices	+	+
Dependable Product	+	+
Change Tolerance	+	- (Ease of Change)
Cost Effectiveness	+	- (Foreign Nationals, COTS)
Deep Community Relations	+	- (Collaborative Research)
Personnel Management	- (Personnel Turnover)	- (Personnel Turnover)
Survivability	- (Resist, Recognize, Reconstitute)	+

Dependable software products are available, reliable, predictable, tested, defect free, fault free, failure free, stable, private, and safe. Well-engineered software products are also change-tolerant and are adaptable, extensible, interoperable, modifiable, and open.

Cost-effective production is driven by a variety of factors involving personnel resources and skills and development environment and its process, methods, and tools. Specifically, there has been a heavy dependence on several approaches, including using foreign nationals and offshore outsourcing, the incorporation of commercial off-the-shelf products, the deepening of community relations through collaborative research, and the management of personnel factors, in particular personnel turnover.

Survivability spans the resistance to cyberattack, the recognition of a cyberattack, and the reconstitution of enterprise software operations following a cyberthreat or cyberattack [12]. Survivability is achieved through the right blend of function, form, and fit. Function includes user authorization, access control, encryption, firewalls, proxy servers, normal operation monitoring, backup and shadow operations, data and program restoration, and disaster recovery. Form includes dispersion of data, diversification of systems, rules of construction, state data isolation, disciplined data, intrusion usage patterns, virus scans, internal integrity, secure state data monitor, exception handlers, full system state architecture, minimum essential function, and isolation of damage. Fit includes adherence to loading limits, predictable response, no memory leaks, rate monotonic scheduling, timeline or event-driven scheduling, monitor memory management, timeline predictability, watch-dog timer, and full system predictability.

Leading indicators are identified for each practice and form the basis for the trade off that is structured along the following lines:

- Engineering practices and dependable product factors enhance both competitiveness and security.
- While change tolerance and ease of change benefit competitiveness, they also provide easy access for those with malevolent intent.
- While cost effectiveness benefits competitiveness, some of the means for achieving it present security exposures. Foreign nationals are skilled and cheap [13, 14]; however, they possess the means in the form of superior knowledge and access to intrude on the nation's critical infrastructure, and they lack allegiance to the United States. Commercial off-the-shelf products provide quick and cheap solutions [15, 16]; however, they are produced with unknown work forces using unknown practices that yield unknown trustworthiness – a security exposure.
- While collaborative research with appropriate intellectual controls is necessary to achieve high maturity in competitiveness, this same knowledge could be used to launch a highly intelligent security intrusion.
- Personnel turnover impacts both competitiveness and security; deep domain knowledge must be kept intramural.
- Survivability practices essential for security impact competitiveness through added cost, product inconvenience, and increased complexity.

The leading indicators (see Figure 3) selected to characterize the practices and factors of competitiveness and security are drawn from the attributes of trustworthy software systems [7], global software competitiveness [17, 18], and cybersecurity survivability [12, 19].

A Web-based scoring and analysis tool is being used to assess the impact of trustworthiness, cost effectiveness, and survivability practices and factors on competitiveness and security. Using this tool, the factor impact analysis was conducted to analyze the behavior of trustworthiness, cost effectiveness, and survivability (see Figure 4). To demonstrate the use of the tool, a set of notional quick-look scores is postulated for commercial, Department of Defense, industry, and government (see Table 2 on Page 28). Participants are asked what scores they would assign each practice and factor and are invited to exercise the tool to complete the analysis [20].

Each practice and factor is rated from low to high on a scale of one to five. The expressions used to evaluate competitiveness and security derives an average of the factors, not weighted. Negative factors shown in Table 1 are adjusted by subtracting the score for the factor from six effectively mapping the one-to-five scale to a five-to-one scale.

The expressions used to evaluate competitiveness and security are:

competitiveness= (engineering+dependable+change+

Leading Indicators of Competitiveness and Security

Engineering Practice
- Complete
- Correct
- Consistent
- Conforming
- Traceable
- Low Complexity
- Scalable
- Predictable
- Usable

Dependable Product
- Available
- Reliable
- Predictable
- Tested
- Defect Free
- Failure Free
- Fault Free
- Stable
- Private
- Safe

Change Tolerant
- Adaptable
- Extensible
- Interoperable
- Modifiable
- Open

Foreign Nationals and Outsourcing
- Immigration Policy
- Domestic Outsource
- Offshore Outsource

Commercial Off-the-Shelf
- Reuse Technology Practice
- Product Line Practice
- Domain Architecture

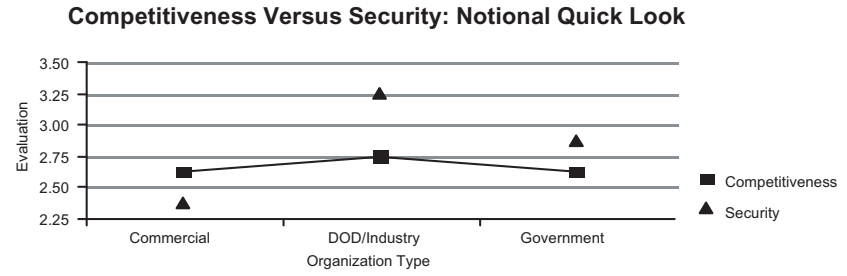
Deep Community Relationships
- Collaborative Research
- Government Research
- University Research

Personnel Management
- Open Requisitions
- Personnel Turnover
- Staff Churn

Survivability
- Resistance
- Recognition
- Reconstitution

Figure 3: *Leading Indicators*

Figure 4: *Competitiveness Versus Security Trade off*



foreign+cots+research+
(6-personnel)+(6-survivability))/8

security=
(engineering+dependable+
(6-change)+
(6-foreign)+(6-cots)+(6-research)+
(6-personnel)+survivability)/8

While both are essential, it is clear that competitiveness and security travel on separate paths that do crisscross and overlap at certain points. This competitiveness versus security trade-off may be tilted toward competitiveness thereby exposing the nation's critical infrastructure to predictable security threats.

Survivability

The nation's software infrastructure is fragile. When it is targeted by a competent, determined attacker, it may collapse. Those who bring their A-game may be able to reconstitute software operations; others will not.

Survivability spans the resistance to cyberattack by improving the software infrastructure, recognizing a cyberattack by sharing information on threats and vulnerabilities, and reconstituting enterprise software operations following a cyberthreat or cyberattack by ensuring continuous operations, switching over, and restarting critical operations. Survivability is achieved through the right blend of function, form, and fit (see Table 3).

The game plan is a software survivability policy that begins by forging a shared vision on the nature of the threat, vulnerabilities, and readiness. This vision assumes that threats continuously evolve, vulnerabilities are large and growing, critical assets are under continuous attack by insiders and outsiders, attacks are targeted and persistent and directed at both system and application, threats and vulnerabilities are outside the control of the enterprise and not fully knowable, and survivability strategies must be independent of threats and vulnerabilities.

The policy establishes a readiness framework for achieving software survivability, one that organizes and orchestrates the layers of security by making an explicit commitment for inaction or action based on security costs exceeding intrusion costs, adopting best security practices in order to avoid lawsuits, performing due diligence in resistance and recognition in order to protect the business enterprise, ensuring the continuous operation of the critical infrastructure through reconstitution, and controlling the disclosure of information to the government and to attackers (see Table 4).

Practice	Commercial	DoD/Industry	Government
Engineering	1	3	2
Dependable Product	2	3	1
Ease of Change	2	3	1
Foreign Nationals	4	2	3
Commercial Products	4	2	2
Collaborative Research	2	4	3
Personnel Management	4	3	2
Survivability	2	4	1

Table 2: *Notional Values for Factors*

Conclusion

When it comes to security, knowledge must replace both power and money as the coin of the realm. Both government and industry have responsibilities to reconcile the conflicting factors encountered in seeking both competitiveness and security. While the government cannot make us safe from cyberattack, it can tilt the business calculation toward security through tax credits, insurance mechanisms, and selective indemnification designed to incentivize readiness. Since the industry's software products make us vulnerable to cyberattack, industry must make the sacrifices needed to achieve security by rebalancing its cost effectiveness tactics and ensuring the readiness and survivability of software products.◆

References

1. Information Security Alliance. Common Sense Guide for Senior Managers - Top Ten Recommended Information Security Practices. 1st ed. Arlington, VA: Information Security Alliance, July 2002 <www.isalliance.org/news/requestform.cfm>.

Table 3: *Software Survivability Model*

	Function	Form	Fit
Resistance • Bulletproof	• User Authorization • Access Control • Encryption • Firewalls • Proxy Servers	• Dispersion of Data • Diversification of Systems • Rules of Construction • State Data Isolation • Systematic Programming • Disciplined Data	• 50 Percent Loading • Predictable Response • No Memory Leaks • Rate Monotonic Scheduling • Timeline vs. Event Driven
Recognition • Detect	• Cyber Forensics • Normal Operation Monitoring • Backup Operation • Shadow Operation • Fully Redundant Operation • Voting	• Intrusion Usage Patterns • Virus Scans • Internal Integrity Checking • Secure State Data Monitor • Exception Handlers	• Monitor Memory Management • Timeline Predictability • Watchdog Timer
Reconstruction • Restore • Continue	• Restore Data and Programs • Minimum Essential Function • Alternative Services • Disaster Recovery	• Full System State Architecture • Minimum Essential Function • Isolation of Damage	• Full System Predictability • Reduced Volume • Conserve Time and Memory

Table 4: *Software Survivability Policy*

Policy Step	Enterprise Objective	Leading Indicators
Commitment to • Inaction • Action	• Understand the Costs	• Security Costs • Intrusion Costs
Adopt Best Practices	• Avoid Lawsuits	• Culture of Security • People Doing the Protecting • Personnel Background Check
Perform Due Diligence	• Protect Business	• Resistance • Recognition • Cost Effective Sacrifices
Ensure Continuous Operation	• Protect Critical Infrastructure	• Reconstruction • Architecture Sacrifices • Change Tolerance Sacrifices
Control Disclosure of Information	• Open to Government • Hidden to Attackers	• Information Sharing With Government • Information Hiding From Attackers

2. Vatis, Michael A. "Cyber Attacks During The War on Terrorism: A Predictive Analysis." Institute for Security Technology Studies at Dartmouth College, 2001.
3. O'Neill, Don. "Competitiveness Versus Security Assessment Tool." <http://members.aol.com/ONeill Don2/comp-sec_frames.html>.
4. Software Engineering Institute. "Process Maturity Profile, Software CMM, CBA-IPI, and SPA Appraisal Results, 2003 Mid-Year Update." Pittsburgh, PA: Software Engineering Institute, Sept. 2003 <www.sei.cmu.edu/sema/pdf/SW-CMM/2003 sep.pdf>.
5. O'Neill, Don. "Managing Architecture." The Competitor 4.6 (July 2001) <http://members.aol.com/ONeill Don2/competitor4-6. html>.
6. Software Engineering Institute. Architecture Trade-off Analysis Method SM. Pittsburgh, PA: Software Engineering Institute <www.sei.cmu.edu/ata/products_services/atam. html>.
7. O'Neill, Don. "Trustworthiness of Software Value Points." The Competitor 4.3 (Jan. 2001) <http://members.aol.com/ONeillDon2/ competitor4-3.html>.
8. Krebs, Brian. "Most Federal Agencies Flunk Interent Security." Washington Post 10 Dec. 2003.
9. O'Neill, Don. "Software Value Added Study." ACM Software Engineering Notes 22. 4 (July 1997) <http://members.aol.com/oneilldon2/new_competitor_initial.html>.
10. Council on Competitiveness. "U.S. Competitiveness: A Ten Year Strategic Assessment." Washington, D.C.: Council on Competitiveness, Oct. 1996.
11. O'Neill, Don. "Set Direction, Provide Fuel, and Control Environment ... Be Globally Competitive." e-GOV Journal 2.1 (Dec./Jan.1999) <http://members.aol.com/oneilldon2/new_vol1no3. html>.
12. Linger, Richard C., and Andrew P. Moore. "Foundations for Survivable System Development: Service Traces, Intrusion Traces, and Evaluation Models." CMU/SEI-2001-TR-029. Pittsburgh, PA: Software Engineering Institute, Oct. 2001.
13. CXO Media Inc. "A Passage to India." CIO Magazine 1 Dec. 2000 <www.cio.com/archive/120100/index. html>.
14. Moitre, Deependra. "Country Report on India's Software Industry." IEEE Software Magazine Jan. 2001.
15. Basili, Victor R., and Barry Boehm. "COTS-Based Systems Top 10 List." Computer Mar. 2001: 91-93.
16. Software Engineering Institute. COTS Usage Risk Evaluation. Pittsburgh, PA: Software Engineering Institute <www.sei.cmu.edu/cbs/cureprod.htm>.
17. O'Neill, Don. "Global Software Competitiveness Assessment Program." Quality Week Europe Conference, Brussels, 1997.
18. O'Neill, Don. "Global Software Competitiveness Assessment Tool." <http://members.aol.com/ONeill Don2/special_gsc_frames.html>.
19. Software Engineering Institute. Operationally Critical Threat, Asset, and Vulnerability Evaluation. Pittsburgh, PA: Software Engineering Institute <www.sei.cmu.edu/programs/nss/surv-net-mgt.html>.
20. O'Neill, Don. "Competition Versus Security." Quality Week 2002 Conference, San Francisco, CA., Sept. 2002 <http://members.aol.com/ONeillDon2/comp-sec-paper.html>.

Note

1. Dr. W. Edwards Deming is known as the father of the Japanese postwar industrial revival and is regarded by many as the leading quality guru in the United States. He died in 1993.

About the Author

Don O'Neill is an independent consultant who focuses on software inspections training, directing the National Software Quality Experiment, and conducting global software competitiveness assessments. Following a 27-year career with IBM's Federal Systems Division, O'Neill completed a three-year residency at Carnegie Mellon University's Software Engineering Institute under IBM's Technical Academic Career Program. He is a founding member of the Washington D.C. Software Process Improvement Network and the National Software Council and serves as the executive vice president of the Center for National Software Studies. O'Neill is also a collaborator with the Center for Empirically Based Software Engineering.

9305 Kobe Way
Montgomery Village, MD 20886
Phone: (301) 990-0377
E-mail: oneilldon@aol.com

Introducing TPAM: Test Process Assessment Model

Dr. Yuri Chernak
Valley Forge Consulting, Inc.

This article presents a Test Process Assessment Model, TPAM™, that can be used in conjunction with the Capability Maturity Model® (CMM®) Level 2 and Level 3. TPAM is fully consistent with the CMM structure. It presents the test process using three key process areas and defines their process goals and practices.

The Software Engineering Institute's (SEI℠) Capability Maturity Model® (CMM®) [1] has a long and successful history of being used by software organizations for assessing and improving their software process. Another strong trend – offshore software development – has also contributed to the increased use of the CMM. American businesses use this framework as a standard approach to assess and select their offshore partners. Likewise, offshore software development companies, especially in India, use the CMM certification as a marketing tool to promote their services and compete for contracts.

One of the known limitations of the CMM is that it does not sufficiently address the software test process. The few testing-related practices defined by the key process area (KPA) Software Product Engineering at CMM Level 3 do not provide sufficient visibility into the test process capability, nor can they be used as a framework for test process improvement. To fill this void, a number of testing maturity models have emerged since the mid 1990s. Some of them were designed to be used in conjunction with the CMM [2, 3, 4]. However, none of these models has gotten much acceptance so far, which motivates us to continue research in this area.

Even though the SEI's CMM Integration℠ [5] covers the test process much better than its predecessor, a transition from the CMM to the CMMI is not going to happen overnight. Thus, we can expect that U.S. companies, performing either self-assessments or capability evaluations for selecting their subcontractors, will continue using the original CMM for some time. To help these companies assess and improve their test process, this article introduces a Test Process Assessment Model (TPAM™, pronounced tee-pam) that has been developed to be a CMM companion model intended to complement the CMM framework at Level 2 and Level 3. TPAM has been primarily influenced by the Systematic Test and Evaluation Process (STEP) methodology [6]. It has been evolving over the years and reflects the author's experience with large-scale projects delivering critical systems used on Wall Street.

Due to space constraints, CROSSTALK *was not able to publish this article in its entirety. However, it can be viewed in this month's issue on our Web site at <www.stsc.hill.af.mil/crosstalk> along with back issues of* CROSSTALK.

LETTER TO THE EDITOR

Dear CROSSTALK Editor,

In the article "Better Communication Through Better Requirements," CROSSTALK April 2004, Michael Hillelsohn states, "An effective requirement communicates clearly to all parties …" Let's examine the following assumption underlying this article and most current requirements work: One can communicate clearly with all parties (stakeholders) about system needs by relying primarily on natural language (e.g., English, French, etc.) expressions.

I believe this assumption is false.

Natural language has value in introductions and overviews, but is unsuitable for precise or complete specification – even if scrubbed. Its words (components) and sentences (structures) are inherently ambiguous, and its expressions are bulky. To deal with this problem, branches of mathematics (algebra, statistics, calculus) have been developed to express precise relationships about various facets of our world. Mathematics, however, has its own problems with readability.

This suggests that the needs of cost-effective requirements specification might be met with a compromise that blends the familiarity of (well-defined) domain terminology with the structured expressions of mathematics.

As an example, consider the need for functionality to create a valid order. To understand this requirement, you must understand the terms *order*, *valid order*, and *create a valid order*. Order is both a domain entity and a system entity. As a system entity it has attributes, value ranges, and relationships with other system entities (e.g., customers). Valid order is a [hairy] Boolean expression involving attributes and relationships of several entities. Create is an action that should be specified by defining the conditions that are TRUE after a successful create (i.e. post-conditions) that are not TRUE before.

Failure to supply these detailed definitions at requirements time, means that they will be supplied later and most likely will not be effectively validated. Therefore, the system must fail in test or production to reveal misunderstandings or omissions. Precision always happens (e.g., code), if a working system is produced. The issue is not if precision, but when it first appears, who provides it, and when it can be validated.

David Gelperin
LiveSpecs Enterprises
david@livespecs.com
Web Site: <www.livespecs.com>

I'm Sorry, Dave – I Can't Certify That

Come on, you know the reference I am making in the above title, right? Remember back in 1968 when the movie "2001: A Space Odyssey" came out? Based on the book by Arthur C. Clarke, this Stanley Kubrick movie was truly awesome! The HAL 9000 [Heuristically programmed ALgorithmic] computer was superb! (Did you ever notice that HAL is one letter below IBM in the alphabet?)

The HAL computer, voiced by the actor Douglas Rain, was particularly interesting. The computer, which developed paranoia, eventually killed four[1] crew members, and then tried to kill Dave Bowman, who was trapped outside of the ship in a smaller space pod without his space helmet. He asks the HAL computer to open the pod bay doors to let him back inside the ship, and HAL says "I'm sorry, Dave – I can't do that!" (As a side note, imagine the home life of Douglas Rain. His wife would ask him to do something simple, like "Honey, could you carry the trash out?" He would reply "I'm sorry …" and his wife would probably run screaming from the room.)

In a later novel (and movie), "2010: Odyssey Two," it was explained that HAL was given conflicting orders that drove the poor computer into a psychotic mania, forcing him to try to kill all of the crew. Imagine that – conflicting orders drove the computer crazy. Gee – wonder what that would do to a developer?

Back in the 1940s, John von Neumann originated the idea of what we consider modern computer architecture[2]. It included basic concepts such as data/instruction store, a central processing unit, input/output, etc. The nice thing about hardware was that visualization techniques (blueprints, circuit diagrams) allowed developers to see what they were building.

By the mid-60s, hardware technology had advanced so rapidly that software, not hardware, was becoming the limiting factor. Various methods were used to help developers visualize the software. One of the early tools – still used today – was a thing called a HIPO (Hierarchical Input Process Output) chart, which displayed a top-down visualization of the major components in a system.

I remember when I first learned how to use the HIPO process. In fact, I still have my original IBM-supplied green HIPO template. It seemed so easy: consider the inputs to your system, develop processes to manipulate the data, and produce output. How hard could that be? As any seasoned developer will tell you, it can be very hard!

When I was learning the HIPO process, I was given toy problems such as "Given three sides, determine the area of a triangle." Eventually, you would realize that a better problem statement was, "Given three sides, determine if they are indeed a triangle, and calculate the area of the triangle." The point eventually driven home was that you cannot ever trust the input.

Then it was time for another new law: "Garbage In – Garbage Out" (GIGO), that says that given garbage as an input, you should expect garbage as an output.

However, GIGO has recently taken on a new meaning. I have seen it referred to as "Garbage In – Gospel Out." This meaning implies that we poor humans have a tendency to implicitly trust the output of a computer. In other words, it doesn't matter how good the input was or how incorrect the process was, the output is considered gospel truth.

And that leads us back to the title of this column. How do you really know that your output is good? By making sure that the process is correct, and that the input data is valid. You do this by the process of verification, validation, and accreditation (VV&A). And part of the VV&A process consists of looking at the quality of the process.

Now, I am not saying that being Capability Maturity Model® Integration (CMMI®) Level X or ISO 9001 guarantees that you are producing top-quality software. I'm also not saying that having an assessment of your processes will make things better. What I am saying is that being CMMI Level X or ISO 9001 gives me a great deal more confidence that at least you care, and having an assessment lets me know what my weak spots are. Have you ever bought a used car? Did you have a good mechanic check the car out? Did he guarantee that the motor would not fall out after 10 miles, or that the transmission would not fail? Nope. But he did guarantee that basic things appeared OK, which gave you a bit more faith in the reliability of the vehicle. You were more willing to spend your money. And, if the mechanic said, "The car is OK – but you need new front shocks," you knew what you needed to fix in the short term.

Software development is much the same. Don't you feel a bit more confident spending your money knowing that the basics are all covered? Having a few experts tell you where your process is deficient, and letting you know what to fix first saves money and lowers anxiety.

As software and system developers, we are often given conflicting orders, just like HAL. "Cut costs – but keep quality up!" "Add these requirements, but don't increase the delivery schedule!" "Keep full functionality, but cut 10 percent from your development budget!" Perhaps, instead of carrying bottles of Valium and Prozac to my next budget and schedule meeting, I can use assessments and certifications to help me make the best of what I have, and improve what I have left to work with.

Just like HAL, I sometimes have a tendency to get a bit paranoid. While I have yet to reach the point HAL reached, I *have* sometimes considered locking a few managers and co-workers[3] outside the building to improve productivity.

— David A. Cook, Ph.D.
Senior Research Scientist
The AEgis Technologies Group, Inc.
dcook@aegistg.com

1. Well, actually only three. In a later book by Clarke (3001: The Final Odyssey), Dr. Frank Poole was found floating in space about a thousand years later, and successfully revived.
2. John von Neumann, et. al. "Preliminary Discussion of the Logical Design of an Electronic Computing Instrument." 1946 <www.cs.unc.edu/~adyilie/comp265/vonNeumann.html>.
3. For all of my current and former co-workers and managers, were you expecting to see your name here?

S T S C

Published by the Software Technology Support Center (STSC)

CrossTalk / MASE
6022 Fir Ave.
Bldg. 1238
Hill AFB, UT 84056-5820

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