FAQs: An Introduction to Software Product Lines

NEWS AT SEI

Author

Paul C. Clements

This library item is related to the following area(s) of work:

Software Product Lines

This article was originally published in News at SEI on: March 1, 2005

Organizations that leverage core assets (software, designs, documentation, test artifacts, budgets and schedules, tools, and more) across the family of systems they produce (instead of building each system separately) are able to dramatically increase quality and reduce cost and time to market. But adopting a product line approach to software is a technical and a business decision that involves many challenges. Over the next few issues, this column will answer some of the most frequently asked questions of organizations considering a product line approach.

"Isn't a product line just the group of products produced by a single business unit or profit/loss center?"

That is what some people mean when they use the term, but it's not what we mean. We define a software product line as **a set of software-intensive systems that share a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way**. A product line is a set of products carefully chosen to take simultaneous advantage of commonality and market opportunities. These opportunities may or may not span what is normally produced by a single business unit. For example, the software for commercial avionics and the software for military avionics could span separate business units in a company but be developed as a single product line by a software group that reports to both. Or, a single business unit may be responsible for developing and fielding more than one software product line. In general, a business unit is formed for organizational or financial reasons and may be responsible for (part of) one or more product lines.

Related Links

Training

Software Product Lines - eLearning

Software Product Lines

See more related courses »

"If a product line is a set of products, does that mean I have to build them all?"

No. While the term "product line" usually brings to mind a specific set of fielded products, you can think of a product line as defining a virtual set of products, each sharing a set of common, managed features and able to be built from the same core asset base. Only those products that satisfy some specific market need are actually built. The others represent untapped production capability that could be built should the opportunity arise.

"What's the difference between a domain and a product line? For instance, there is a telecommunications domain and some companies build a telecommunications product line. What's the difference?"

A domain is a specialized body of knowledge, an area of expertise, or a collection of related functionality. The telecommunications domain is a set of telecommunications problems, which in turn consists of other domains (switching systems, protocols, telephony, networks, etc.). A telecommunications product line is a specific set of systems that addresses some of those problems. People often use the term "domain" incorrectly to refer to a set of systems that employ knowledge in a particular domain, but that's an incorrect use of the term. It's hard to imagine any non-trivial software system that doesn't encompass knowledge from several domains.

"Isn't software product line practice the same as single-system development with reuse?"

No. It differs in three fundamental ways. First, building a software product line is about planning a plurality of products that will exist in the field and be maintained simultaneously, not just one that evolves over time. Second, the reuse that's involved is carefully planned, strategic, and applicable across the entire set. In fact, one of the primary distinguishing characteristics of product line practice is planned reuse rather than ad hoc reuse. Software product line practice encourages choices and options that are optimized from the beginning for more than a single system. And third, single-system development with reuse usually begins by taking the latest version of whatever other systems happen to exist, copying the code, and starting to make product-specific changes to it. The result is that each version of each product launches on its own maintenance and change trajectory, which soon overwhelms the organization trying to keep all of the versions under control and staffing the effort. In a software product line, the core assets follow a single evolution path, and all versions of all products are built from those.

"Vendors and other developers issue subsequent releases of single products all the time, and have been doing so for years. Each release is built in the same way — you would say from the same core asset base. Technically speaking, what's the difference between a software product line and multiple releases of the same product?"

There are some similarities, but the differences are acute enough to matter. The primary difference is that vendors who release subsequent versions try to retire earlier versions as soon as they can. Also, later versions are usually supersets of earlier versions. The latest version is always the one of interest. In a software product line, all versions are of interest, because they provide different functionality and quality attributes, each serving different market segments or missions.

"Isn't product line practice just another name for domain engineering?"

Domain engineering addresses only half of the problem (core asset development and acquisition). Product line practice addresses both domain engineering and product development using the core assets, or what has been called application engineering.

"Isn't product line practice just another name for component-based development?"

Software product lines certainly rely on a form of component-based development. Component-based development as it is typically defined involves the selection of components from a library or the marketplace to build products. Though the products in software product lines certainly are composed of components, these components are all specified by the product line architecture. Moreover, the components are assembled

in a prescribed way; the prescription comes both from the architecture and the production plan. Software product lines involve the strategic use of components. Component-based development usually is missing such a strategic and systematic approach.

A Note on Terminology

In this column we use the following terms:

A *software product line* is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way.

Core assets are those reusable artifacts and resources that form the basis for the software product line. Core assets often include, but are not limited to, the architecture, reusable software components, domain models, requirements statements, documentation and specifications, performance models, schedules, budgets, test plans, test cases, work plans, and process descriptions. The architecture is key among the collection of core assets.

Development is a generic term used to describe how core assets (or products) come to fruition. Software enters an organization in any one of three ways: the organization can build it itself (either from scratch or by mining legacy software), purchase it (buy it, largely unchanged, off the shelf), or commission it (contract with someone else to develop it especially for them). So our use of the term "development" may actually involve building, acquisition, purchase, retrofitting earlier work, or any combination of these options. We recognize and address these options, but we use "development" as the general term.

A *domain* is a specialized body of knowledge, an area of expertise, or a collection of related functionality. For example, the telecommunications domain is a set of telecommunications functionality, which in turn consists of other domains such as switching, protocols, telephony, and network. A telecommunications software product line is a specific set of software systems that provide some of that functionality.

Software product line practice is the systematic use of core assets to assemble, instantiate, or generate the multiple products that constitute a software product line. The choice of verb depends on the production approach for the product line. Software product line practice involves strategic, large-grained reuse.

Some practitioners use a different set of terms to convey essentially the same meaning. In this alternate terminology, a *product line* is a profit and loss center concerned with turning out a set of products; it refers to a business unit, not a set of products. The *product family* is that set of products, which we call the product line. The core asset base is called the platform. Previously, what we call core asset development was often referred to as *domain engineering* and what we call product development was referred to as *application engineering*.

The terminology is not as important as the concepts. Readers might encounter different sets of terms in other places and should be able to translate between them.

About the Author

Paul Clements is a senior member of the technical staff at the SEI, where he has worked for 10 years leading or co-leading projects in software product line engineering and software architecture design, documentation, and analysis. Clements is the co-author of three practitioner-oriented books about software architecture: *Software Architecture in Practice* (1998; second edition, 2003), *Evaluating Software Architectures: Methods and Case Studies* (2001), and *Documenting Software Architectures: View and Beyond* (2002). He also co-wrote *Software Product Lines: Practices and Patterns* (2001), and was co-author and editor of *Constructing Superior Software* (1999). In addition, Clements has also written dozens of papers in software engineering reflecting his long-standing interest in the design and specification of challenging software systems. He received a BS in mathematical sciences in 1977 and an MS in computer science in 1980, both from the University of North Carolina at

Chapel Hill. He received a PhD in computer sciences from the University of Texas at Austin in 1994.

The views expressed in this article are the author's only and do not represent directly or imply any official position or view of the Software Engineering Institute or Carnegie Mellon University. This article is intended to stimulate further discussion about this topic.

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

Improving Defense Software-Intensive Systems Acquisition with CMMI-AM

NEWS AT SEI

Authors

Brian P. Gallagher

Sandra Shrum

This library item is related to the following area(s) of work:

Acquisition Support

CMMI

This article was originally published in News at SEI on: January 1, 2005

Related Links

News

SEI Cosponsors Agile for Government Summit

Training

Data Rights and DoD Acquisition

SGMM Navigator Training

See more related courses »

The goal of acquiring software-intensive systems in the defense industry is to deliver weapons systems cost effectively and quickly to the warfighter. However, the quality of defense systems in general, and software intensive defense systems in particular, results from the processes used to create and maintain them. Therefore, improving acquisition processes directly benefits warfighters by providing enhanced capabilities and technologies quickly to the field.

Acquirers Face Complex Challenges

Unfortunately, the increasing complexity of defense weapons systems has overtaken the experience and capabilities of the organizations acquiring them. As a result, many acquirers are finding it difficult to establish robust systems engineering practices within the program office, stabilize requirements enough to adequately work with developers/suppliers, and estimate the time and effort required to deliver a capability or system.

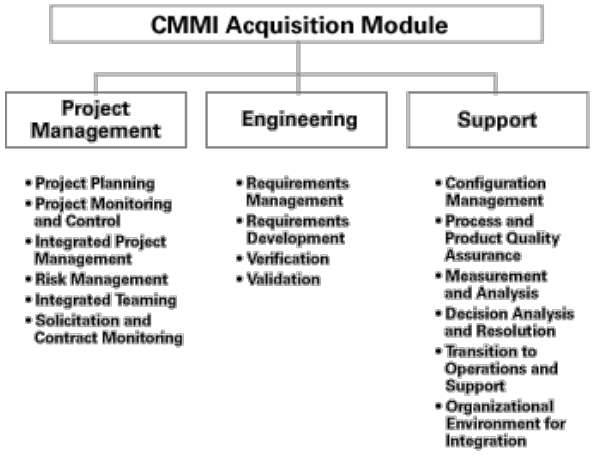

Figure 1: The CMMI Acquisition Module and its relevant process areas

Defense system acquirers are also finding it difficult to enforce schedule milestones and delivery, assess the technical risk involved in acquiring particular products, implement process control measures, track short- and long-term costs in relation to a budget, and continuously identify and mitigate risks in a team environment.

Overcoming these challenges requires a new set of capabilities. Acquirers must understand the operational context of the effort. They must codify the desired functionality into a system that can be implemented. They need to continuously evaluate both the system and the development teams' ability to meet evolving requirements that reflect changes in timing, cost, and function. They also must identify the risks involved in selecting one development team or set of suppliers over another, and proactively collaborate with the team to ensure that risks are identified and mitigated. In essence, organizations must move from ad hoc acquisition to repeatable and measurable acquisition practices. The Software Engineering Institute (SEI) is helping acquirers to obtain these skills and capabilities by developing the Capability Maturity Model Integration Acquisition Module (CMMI-AM).

Helping Defense Software-Intensive Systems Acquirers Improve their Processes

A collaboration of the Department of Defense (DoD), industry, and the SEI, the Capability Maturity Model Integration (CMMI) infrastructure integrates best practices from systems engineering (SE), software engineering (SW), integrated product and process development (IPPD), and supplier sourcing (SS). The CMMI Acquisition Module identifies defense software intensive system acquisition practices that complement CMMI.

For example, CMMI-AM practices are focused both inside the acquiring organization to ensure that the acquisition is being conducted effectively, and outside to help the organization conduct project monitoring and supplier oversight. At the same time, CMMI-AM best practices provide a foundation discipline and rigor that enable products and services to be acquired successfully. In essence, the CMMI-AM presents a structure that helps acquirers examine the effectiveness of their processes, prioritize improvement goals, and achieve them.

During the summer of 2004, SEI researchers piloted the CMMI Acquisition Module with several DoD programs. In these pilot studies, participants evaluated the effectiveness of their acquisition activities, and used the module to establish process improvement programs that comply with Section 804 requirements. (See the sidebar for information on Section 804.) This piloting activity is sponsored by Dave Castellano, deputy director, Systems Engineering, Defense Systems, Office of the Secretary of Defense, Acquisition, Technology, & Logistics (AT&L).

Once the pilot studies have been reviewed, an updated module will be made available for use. However, it is not too early to learn more about CMMI and CMMI-AM.

Government Promotes Acquisition Improvement

The U.S. government has shown its desire to improve the state of acquisition practice in Section 804 of the National Defense Authorization Act.1 This section states, "Service/departments shall establish programs to improve the software acquisition process." The requirements of such a program include

- a documented process for planning, requirements development and management, project management and oversight, and risk management
- metrics for performance measurement and continual process improvement
- a process to ensure adherence to established process and requirements related to software acquisition

The act also requires that the Office of System Architecture and Investment Analysis (Communications, Command, Control, and Intelligence) and the Undersecretary of Defense AT&L support government programs by

- prescribing uniform guidance for improving software intensive systems acquisition programs across the DoD
- assisting the services and departments by ensuring that source selection criteria include past performance and the maturity of the software products offered by potential sources; and serving as a clearinghouse for best practices in software development and acquisition in both the public and private sectors

[1] U.S. Congress. "National Defense Authorization Act for Fiscal Year 2002." Calendar No. 163, 107th Congress, 1st Session, S. 1438. Washington, D.C., 2001.

Enterprise Security Management: Refocusing Security's Role

NEWS AT SEI

Author

Richard A. Caralli

This article was originally published in News at SEI on: January 1, 2005

As their understanding of security deepens, organizations are looking for new techniques to manage security. Enterprise security management (ESM) is an emerging research area that aims to help organizations in this quest for a better approach to security by making it an enterprise-wide competency. This means a fundamental shift in focus as an organization ensures that security is used to achieve its goals and become more resilient, while managing security as a core competency instead of as an extension of information technology.

In an outgrowth of fieldwork deploying information security risk assessment methodologies, a team in the Networked Systems Survivability program at the Software Engineering Institute (SEI), consisting of Rich Caralli, James Stevens, Bradford Willke, and Bill Wilson, is identifying and examining the core capabilities that define a framework for security management in an organizational and operational context. In this context, the practice of security is viewed as an activity that keeps the organization's productive elements—people, assets, and technology—free from harm or disruption so that they can perform their intended functions and help the organization accomplish its mission.

Managing for Enterprise Security

When managing for enterprise security, an organization cannot separate security from its other goals. Making security one of the business goals of an organization elevates that organization's ability to realize an effective security approach. With enterprise security, the focus expands to all of the processes of an organization. In addition, if an organization is aware of and supports its core competencies, security benefits and protection of critical assets and processes will result even though security is not the main focus.

In developing the ESM approach, the team is concentrating its efforts on several key processes.

- Identify core capabilities The immediate focus of the ESM research is to develop a capabilities framework that represents the essential capabilities necessary for addressing security as a business problem. This framework is intended to document and describe the core capabilities necessary for a systematic, managed, and measured process for securing the assets and processes of medium to large organizations.

  The team studies real-life organizations as they confront various aspects of security management and interact with high-performing organizations that are achieving security effectiveness through expanded focus on other organizational functions such as IT operations and asset management. The goal is to identify common core capabilities that can be used across organizations. These core capabilities
  - do not always have security as their main focus
  - represent many of the horizontal competencies that organizations already have to conduct business
  - are necessary for organizations to achieve their critical success factors and accomplish their missions
  - are executed throughout the organization, not concentrated in any one operational unit or department
  - are both strategic and tactical in nature

- Develop the appropriate tools, techniques, and methods. Throughout this work, the ESM team continues to develop additional supporting tools, techniques, and methods that facilitate an enterprise approach to security management. One of these methods—the Critical Success Factors Method—has already been documented in an SEI technical report.1 This method helps organizations to establish a foundation for ESM by identifying the organization's strategic drivers and using them to guide security strategies.
- Leverage best practices. In their work with ESM, the team studies multiple sets of best practices to synthesize them and determine what capabilities are being recommended to organizations. The SEI is in a unique position to take a critical look at best practices and to identify the capabilities that they represent. These best practices are beyond the scope of security and relate to many different organizational capabilities. Some of the practices used span the topics of security, IT operations and service delivery, and compliance and regulations. They represent administrative/managerial, technical, and operational practices.

The team has found that by virtue of employing best practices prescribed in some methodologies, some organizations are implicitly covering their security needs while attending to the overall needs of the organization. For example, the practices prescribed in CobiT2 and ITIL3 methodologies easily translate to potential improvements in security and resiliency: if an organization's configuration and change management capabilities are performed consistently and with high quality, many of the vulnerabilities that exist become less of a threat because software updates are regularly installed.

Plans for ESM

ESM continues to grow and be shaped by lessons learned in fieldwork with customers who are trying to improve their security effectiveness and success. The team is currently developing more analysis methods to allow organizations to assess their capabilities for enterprise security management and is actively looking for collaborators who are interested in improving the effectiveness and efficiency of their security efforts at the enterprise level. Potential collaborators include—but are not limited to—large banks, insurance companies, manufacturing organizations, large county and state governments, as well as Department of Defense agencies. The ESM team welcomes interaction with this community so that the emerging capabilities framework represents the best cross section of organizations that are using security as

a way to accomplish their goals and mission.

University Hubs Help SEI Spread Information Assurance Curricula and Methods

NEWS AT SEI

This article was originally published in News at SEI on: January 1, 2005

The Software Engineering Institute (SEI) wants to educate students and professionals around the country in the field of information assurance, but it will need some help to do so.

“We’re a small organization. What we need to do is leverage, in many forms, the infrastructure, capabilities, and relationships that are already there” says the SEI's Carol Sledge, project manager for survivability and information assurance curriculum and educational outreach.

One way to boost the SEI's effectiveness is through partnerships with institutions that serve as hubs for universities and community colleges in a region. This approach is helping dozens of institutions in four geographical regions improve the information assurance content in their curricula and the abilities of faculty to teach information assurance:

- Hampton University in Hampton, VA, is a hub educational transition partner working with the SEI to build the capacity to teach information assurance at 22 colleges and universities in Virginia, North Carolina, Maryland, Delaware, and Washington, DC.
- California State Polytechnic University, Pomona (known as Cal Poly Pomona), and Mount San Antonio College (Mt. SAC) in Walnut, CA, are hub partners that help build information assurance capacity at 23 universities in the Cal State system and 109 community colleges in California.
- Texas A&M, Corpus Christi is a new hub partner working to build capacity at community colleges and universities in Texas.
- Moraine Valley Community College in Palos Hills, IL, is the base for a new National Science Foundation (NSF) Regional Center for Systems Security and Information Assurance (CSSIA). The center includes seven partner institutions that offer information assurance training in Illinois, Minnesota, Michigan, Wisconsin, and Ohio. The center collaborates with some 75 other colleges and universities nationally.

The hubs in the mid-Atlantic, California, and Texas are examples of how the SEI is building on existing relationships, which began when representatives of the hub institutions participated in an NSF-funded program at Carnegie Mellon University. The annual program, which started in 2002, provides educational resources to historically black colleges and universities and Hispanic-serving institutions. The program equips faculty members to teach courses in information security. Several SEI technical staff serve as faculty during the month-long program.

"As a result of this training, the department of computer and mathematical sciences is starting a new option on information assurance for graduate students in computer science," says Mario Garcia, associate professor of computer science at Texas A&M, Corpus Christi.

Robert A. Willis, Jr., chair of the Department of Computer Science at Hampton University and one of the originators of the regional approach, credits the first information assurance symposium held at Hampton in February 2004 with building momentum in the mid-Atlantic. The SEI co-sponsored the symposium and several SEI technical staff members were presenters.

"Faculty who attended the first symposium were excited about the program and some have begun collaborations with faculty from other institutions. As the program continues, individual institutions will be better prepared to offer programs in information assurance and to start the process of certification," Willis says.

In California, "the SEI's influence is fueling an already existing collaboration between Mt. SAC and Cal Poly Pomona with valuable information assurance resources such as source experts, curriculum, and relationships," says John Blyzka, a computer information systems professor at Mt. San Antonio. The two schools are extending that strong, collaborative model to other institutions, he says.

The SEI's curriculum has also been used in small business development workshops hosted by Mt. San Antonio's Small Business Development Center. Daniel Manson, a computer information systems professor at Cal Poly Pomona, adds that the relationship with the SEI is assisting Cal Poly Pomona in its goal to become a National Center of Excellence in Information Assurance Education.

The well established courses of the SEI's CERT Coordination Center (CERT/CC) provide excellent source material for building capacity through the regional hubs. CSSIA, for example, will adapt materials from three courses from CERT/CC into three CSSIA courses. "This is a great opportunity for the SEI to transition its work in information assurance," Sledge says. "The faculty at CSSIA are very knowledgeable, they have funding and backing from the NSF, and they have an excellent plan in terms of how to develop the materials, transition them to the faculty, and then have a second level of transition to other faculty in the regional area."

Erich Spengler, director of CSSIA, says increased quality and credibility are two important benefits. "The partnership greatly increases the community and technical college system's ability to respond to the challenges of adapting, disseminating, and delivering quality, industry-recognized information assurance and cyber security curriculum to students and faculty."

Sledge is pursuing other opportunities in higher education to transition SEI curricula and methods, including seminars and courses at schools of business. "People are realizing that information security is pervasive, from the board of directors down through the entire business enterprise," she says. "You don't just tack it on at the end."

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

An Adoption Roadmap for Software Product Line Practice

NEWS AT SEI

Author

Paul C. Clements

This library item is related to the following area(s) of work:

Software Product Lines

This article was originally published in News at SEI on: January 1, 2005

Over the past several years, software product line practice has emerged as an important and viable software development paradigm. Instead of building suites of related software-intensive systems separately, companies of all sizes and in all application areas are learning to build them as a family from a common set of core assets in a prescribed way. Software product line practice has resulted in dramatic (sometimes order-of-magnitude) improvements in time to market, flexibility, productivity, and product quality.

The Product Line Practice (PLP) program at the Software Engineering Institute (SEI) exists to help build and nurture a community of practitioners and researchers interested in improving the state of software product line engineering. Members of the initiative are working to make software product line practice a low-risk, high-payoff undertaking for developing families of related systems. To this end, the SEI maintains a comprehensive Framework for Software Product Line Practice, describing 29 key technical and managerial practice areas for successful creation and sustainment of a software product line sponsors a successful series of international Software Product Line Conferences maintains an active web site where visitors can find and download the latest information about software product lines has defined a number of software product line practice patterns [Clements and Northrop 02], which help organizations carry out practices in a structured, planned, predictable way

But the primary mission of this initiative is to help organizations make the transition to successful software product line practice. An organization that does not know how to

Related Links

Training

Software Product Lines - eLearning

Software Product Lines

See more related courses »

go about this transition is unlikely to succeed without great difficulty and cost. To help organizations adopt software product line practices, the SEI recently defined a roadmap based on the Adoption Factory pattern [Northrop 04]. Adoption Factory is a pattern that describes the entire product line organization. It is a composite of eight other patterns, showing how they are orchestrated to accomplish the overall objective of setting up and maintaining a software product line capability:

- *What to Build*, which helps organizations decide on the set of products to be included in the product line
- *Each Asset*, which shows organizations how to build individual core assets
- *Product Parts*, which shows how to supply the core assets from which products will be built
- *Assembly Line*, which helps organizations establish the production infrastructure
- *Product Builder*, which shows how to produce the individual products in the product line
- *Cold Start*, which prepares the organization for its first product line operation
- *In Motion*, which shows how to keep the product line organization running
- *Monitor*, which shows how to maintain a picture of how well the product line organization is running

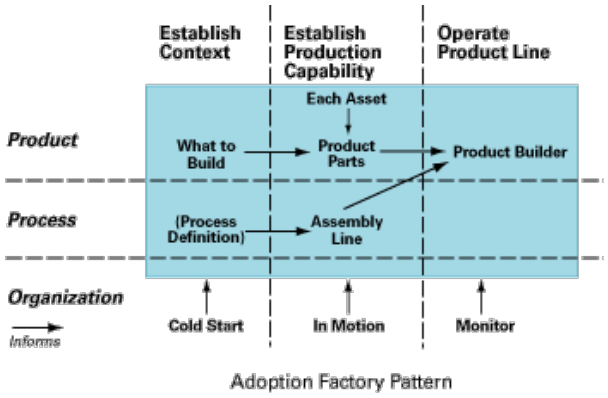

Figure 1: Adoption Factory Pattern

These patterns are in turn defined in terms of the Framework practice areas that each one puts into action and shows how those practice areas are used in concert to produce the desired outcome of each pattern. In addition, Adoption Factory appeals to the "Process Definition" practice area directly, to emphasize an organization's need to define and carry out disciplined processes.

Adoption Factory becomes useful as a roadmap when it is annotated showing

- temporal phases of product line adoption ("establish the context," "establish the production capability," and "operate the product line")
- focus areas on which the patterns and practice areas in Adoption Factory operate (products, processes, and the organization itself)
- the practice areas involved in each part of the pattern
- the artifacts (such as a production plan, or a concept of operations, or a scope definition) produced in each part of the pattern
- the organizational roles carrying the primary responsibility for each part of the pattern

For instance, the figure above shows the pattern annotated with phases and focus areas. This can allow a decision-maker to identify where his or her organization currently is with respect to achieving full software product line capability.

At the SEI, Adoption Factory serves as the unifying theme for several products and services; for example,

- the SEI Product Line Technical Probe and its lighter-weight counterpart the SEI Product Line Quick Look use Adoption Factory to position an organization and characterize its strengths and weaknesses.
- SEI Product Line Planning Workshops use Adoption Factory as the basis to plan specific adoption activities.
- Adoption Factory is the central motif for the Adopting Software Product Lines course, part of the SEI Software Product Line Curriculum and Certificate Program.
- Product Line Analysis, which helps an organization determine the scope and requirements for its software product line, supplies a crucial part of the What to Build pattern component of Adoption Factory.

The PLP Initiative is actively seeking to assist organizations wishing to embrace the software product line approach. A structured, methodical, step-by-step adoption strategy is the cornerstone for a successful transition, and Adoption Factory provides the blueprint.

References

[Northrop 04]
L. Northrop. Product Line Adoption Roadmap (CMU/SEI-2004-TN-035). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2004.

[Clements and Northrop 02]
P. Clements and L. Northrop. *Software Product Lines: Practices and Patterns*, Boston, MA: Addison Wesley Longman, 2002.

CMMI—A Progress Report

NEWS AT SEI

Author

Mike Phillips

This library item is related to the following area(s) of work:

CMMI

Process Improvement

This article was originally published in News at SEI on: January 1, 2005

My last column focused on where the CMMI product suite is headed for the next version. Since then, a SCAMPI Lead Appraisers and CMMI Instructors Workshop and the CMMI Technology Conference and User Group have provided insights on a variety of topics that warrant further discussion in this issue. I've organized the topics by the session where the topic was most specifically addressed.

SCAMPI Lead Appraisers and CMMI Instructors Workshop

Progress on the New Introduction to CMMI (Staged and Continuous) Course

There was a beta delivery of the new Introduction to CMMI (Staged and Continuous) course at the SEI in September. At the SEI Partner Workshop in October, SEI Partners were given a summary of the differences between this new course and the Introduction to CMMI courses that cover only one representation. Authorized Introduction to CMMI instructors who attended the workshop's Instructor Upgrade Course were the first to be approved to offer the new course.

Feedback from these instructors provided a needed adjustment in the schedule for offering the single-representation Introduction to CMMI courses. The new course is designed to be used with the Addison-Wesley book, CMMI: Guidelines for Process Integration and Product Improvement, which covers both representations. Partners believed that using the book may be preferable in some locations, but may be difficult in other locations. Since this concern must be addressed before use of the book can be assured for all course offerings, SEI Partners will continue offering the staged or the continuous courses until this difficulty is resolved. Partner contracts have already been

Please note that current and future CMMI research, training, and information has been transitioned to the CMMI Institute, a wholly-owned subsidiary of Carnegie Mellon University.

Related Links

News

SEI to Co-Sponsor 26th Annual IEEE Software Technology Conference

SEI Cosponsors Agile for Government Summit

Training

See more related courses »

Events

Team Software Process (TSP) Symposium 2014

Nov 3 - 6

modified to allow the delivery of all three courses through 2005. By late FY06, the V1.2 Introduction to CMMI course update will be available, as will an SEI technical report that includes both representations.

The Introduction to CMMI (Staged and Continuous) course was first offered publicly before the CMMI Technology Conference and User Group in Denver, Colorado. Feedback from that offering was positive and reflected a growing recognition that the two concepts of improvement are are complementary. (There's more on the complementary aspects of the staged and continuous representations later in this column.) As we make the transition to the new course, we will continue to offer opportunities for authorized instructors to attend upgrade training that covers the differences between the new course and the single-representation courses. Beginning in February, all new instructors will be trained in the Introduction to CMMI (Staged and Continuous) course, as that is our direction for the future.

The SEI Code of Professional Conduct

The importance of professional standards of behavior in the delivery of SEI products and services has led to the creation of a Code of Professional Conduct. The SEI Partner Workshop allowed for extensive discussions on this topic with the SEI Partners responsible for both CMMI training and appraisals. The Code of Professional Conduct applies to all SEI Partners, SEI Authorized and Certified Professionals and Candidates, as well as SEI personnel who deliver any of the products and services available through the SEI Partner Network. (Technologies such as TSP/PSP and OCTAVE are offered as part of the SEI Partner Network as well as CMMI.) SEI Partner contracts now include a commitment to the Code of Professional Conduct for all SEI Partners. All SEI-Authorized CMMI instructors, SCAMPI Lead Appraisers, and candidate instructors and lead appraisers have committed to institutionalizing this professional standard. The Code can be viewed at http://www.sei.cmu.edu/partners.

SCAMPI and ISO

SCAMPI Lead Appraisers often find themselves assisting organizations that must also measure performance against various ISO standards. Two of the ISO standards that were discussed at the workshop were ISO 9001:2000 and ISO 15504. In the first case, organizations who are committed to quality improvement wish to increase the value of the appraisal effort by satisfying both CMMI and ISO 9001:2000 in a combined effort. Opportunities to explore joint appraisals, recognizing the specific needs of both ISO certifying bodies for ISO audit certificates and the SEI for authorized SCAMPI appraisals were discussed.

The ISO 15504 standard requires both appraisal methods and improvement models. Here the focus of effort is to assure that CMMI performance measured by SCAMPI appraisals meets the ISO 15504 requirements so that the results can be used in development domains such as the automotive industry. Currently the European automobile industry is considering an approach called AutoSPICE. Some companies in that industry, companies using the CMMI Product Suite as their process improvement approach, want to be sure that their efforts are understood in comparison with ISO 15504 results using other models. The potential for pilot appraisals was discussed on this theme. Attendees expressed the wish to be sure that there are no impediments that would prevent SCAMPI appraisals, using the CMMI model, to depict results that satisfy the 15504 standard, preferably in organizations seeking to harmonize their CMMI results with other improvement approaches.

CMMI Technology Conference and User Group

Maturity Levels and Performance

In his lunchtime keynote address to the workshop attendees, Mark Schaeffer, the DoD sponsor for CMMI and Director, Systems Engineering for the Office of the Secretary of Defense, praised the outstanding impact that CMMI provides in broadening the approach for process improvement to a unified approach for all DoD software-intensive systems. However, he also noted that his office sees too many organizations focusing on maturity levels rather than on real improvement: "When achieving a level replaces the focus on continuous improvement, we've lost sight of the goal." He held that appraisal results are too often an end in themselves instead of meeting the government's expectations for actual performance on complex, software-intensive

(< 5 minute) survey.

system development. Schaeffer continued by saying, “The government expects that if you have achieved high maturity, then the next program will perform at that maturity.” He believes that the information available to the acquiring organization seen with the continuous approach – the profile of capabilities against all of the relevant development areas of concern – is more useful than broad statements of organizations with a single-digit maturity-level claim.

Schaeffer indicated that his office would be examining ways of encouraging government use of the continuous (capability-level) appraisal results. As I mentioned above, Mr. Schaeffer's statements are well timed with the development of V1.2 and a “single-book, single-course” approach that assures that the value of the continuous representation is seen by those who may be more familiar with the maturity-level legacy from the Software CMM. Responses from attendees at the first public offering of the new Introduction to CMMI course indicated that a number of the attendees gained a new appreciation for the ability of the continuous representation to enhance capabilities based on business-critical areas as well as the ability of the staged representation to further organizational maturity.

The SEI is providing more information about appraisal results than ever before at http://seir.sei.cmu.edu/pars/. Although the majority of organizations that have agreed to have their information made publicly available (as of this writing, about 130) chose to use a staged appraisal that would result in a maturity level, the depiction of performance is shown for all CMMI process areas.

The continuous appraisals performed differ only in adding the capability level achieved in each of the process areas. (10 of these results have been made publicly available on the PARS site.) For the vast majority of the mid-level appraisals—those emphasizing the maturity level 2 and 3 process areas (i.e., managed and defined)—there are no real differences in the results. The reason for this is that the capability level achieved in each of the process areas matches the staged level requirements for satisfaction of the relevant generic goals. For a staged appraisal, the process area is described as “satisfied.” For a continuous appraisal, the depiction would be “capability level 2” for staged maturity level 2 equivalent appraisals and “capability level 3” for appraisals that are staged at maturity level 3.

Mr. Schaeffer's comments about the continuous approach are similar to those we have been receiving from the Australian Defence Materiel Organization's pioneering work with defense contractors in its country. Its acquisition organization has focused on a broad understanding of capabilities seen within Australia's supply chain rather than seeking maturity levels.

Mr. Schaeffer's points about performance expectations also reflect the SEI's longstanding interest in depicting—and emphasizing—performance results rather than maturity level achievement. Our report on CMMI achievement shows the performance benefits of the commitment to and investment in CMMI-based process improvement. We are also researching better ways to link process capability as measured by appraisal results with performance improvement in projects and organizations.

CMMI Coverage for Service Organizations

In my last column, I described an adjustment to our architectural approach for CMMI models that would allow future “constellations” of CMMI models that address domains often adjacent to the development effort. At the CMMI Technology Conference and User Group, the industry chair of the CMMI Steering Group announced that a team was forming to develop the first set of these related models. The plan is to maximize the common model elements while recognizing the unique practices of—and the differences among—organizations that provide or deliver services and those that develop products and services. We anticipate that the major difference between this new constellation and the existing development constellation will be within the Engineering process areas. If we find that there is terminology that should be adjusted in the existing CMMI development models to assure commonality, we will seek to make any such changes in the V1.2 revisions.

Northrop Grumman Corporation has offered to provide the resources to develop a prototype of the Services constellation in 2005. The development project will be

performed within the existing CMMI Product Team structure. The CMMI for Services team will proceed on a parallel path with the V1.2 effort. We currently envision that a CMMI-Services model could be released a few months after the V1.2 update, currently scheduled for the latter part of FY 2006. I'll be providing more information as this effort takes shape.

Conference Topic Coverage

For the benefit of those unable to attend the CMMI Technology Conference and User Group, I want to provide you with a sense of what was presented over the four days. Tutorials included

- the CMMI's increased coverage of systems engineering
- the expanded family of SCAMPI appraisals
- the power of linking the Team Software Process with CMMI deployment
- moving from the Software CMM to CMMI
- adopting CMMI in small organizations
- using process simulations for continuous improvement
- managing technical people
- balancing agility and discipline

Tracks with multiple presentations included

- CMMI and Process Improvement
- CMMI Extensions
- CMMI for Small Projects and Organizations Appraisals
- High Maturity
- Practical Guidance
- Return on Investments
- Six Sigma
- Systems Engineering
- Tools
- Transitioning to CMMI

Several companies expressed a focus on mission assurance and mission success. This theme crosses CMMI process-area boundaries to encompass process areas such as PPQA, RSKM, SAM, ISM, PI, VER and VAL. This attention is driven by mission criticalities most easily seen with a single missile launch, either to position a satellite in orbit or as part of missile defense. This area appears to be one of growing interest and will provide an opportunity for SEI technical notes or at least a future column dedicated to the topic.

SCAMPI Appraisals for Source Selection and Contract Monitoring

One of the key elements of progress in CMMI appraisals was the DoD's direction to include the external evaluation mode—for source selection and contract monitoring—within the SCAMPI method. This became available with the V1.1 release of SCAMPI, which was accompanied by a guide for use in such benchmark (Class A) appraisals. Earlier approaches, such as the Software Capability Evaluation V3.0, encouraged risk-based evaluations of key areas of concern for a program. With the new Class B and C appraisals, renewed interest in this approach has been evident. Concerns about the SCAMPI Class A approach were twofold: (1) the time required to visit multiple contractor sites and accomplish a full Class A appraisal (typically a staged maturity level 3) would be at least a week for each site, and (2) the maturity level focus, as noted above, was often not well tuned to the program of interest. However, with the interest in the faster, but potentially less rigorous, Class B SCAMPI comes a legitimate concern from the organizations receiving the appraisal for obvious reasons. One approach that merits brief discussion here is a three-phased approach to appraisals.

The first phase is to use a risk based appraisal, as mentioned by Mr. Schaeffer, for the

source-selection element. A subset of CMMI process areas might be chosen for investigation across the proposing contractors. This approach would look for strengths or weaknesses in the development processes that would relate to risks to the proposed development effort. Satisfying CMMI goals, process areas, or maturity or capability levels would not be the point—only risks to the future system's development.

With this approach to source selection, the CMMI Product Suite would become one of the program office's tools for reducing risk in the program, rather than a leveling tool, where all competitors must have a given a maturity level or capability levels to compete. Since levels are not part of the source-selection decision, this approach would encourage process improvement under the CMMI collection of best practices without the "gotta show a level" strategies.

Recall that the greater concerns are about how process discipline will be applied in a new program—not how it was applied in previous ones. Often the new program is an amalgam of multiple contractors at multiple locations with a mixture of experienced professionals and new hires. So the second phase in the appraisal process is to conduct a baseline appraisal once the full team is established. Again, no maturity-level results from any of the contributing contractors are of any particular importance. What is important is the establishment of a clear understanding of the process framework for the program. While this may involve a commitment to common processes, the decision must be considered carefully by the team (i.e., a process trade study). Again, a prudent focus is to use a risk-based analysis of the process strengths and weaknesses, establishing action plans for future checks.

This in turn leads to the third phase, the contract-monitoring phase. This is the periodic checking of progress against the risks catalogued in the baseline phase. Often award-fee provisions can provide desirable incentives for progress. In one prototype use of this approach, the government lead appraiser noted that 41 risks were significantly mitigated within 18 months on a complex multi-contractor DoD intelligence program. On follow-on programs using this approach, some are including the government element of the team in the appraisals.
Note that none of these phases require benchmark appraisals, but they do require healthy customer-supplier relationships and effective contract structures. This represents a mature approach to software-intensive systems acquisition.

Summary

I've sought to give you some insights into a wide variety of the recent activities involving elements of the CMMI Product Suite. Please continue to join the 50,000 folks who visit our CMMI Web site at each day as we seek to continuously improve these tools for your use!

About the Author

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

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

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

An Introduction to Governing for Enterprise Security

NEWS AT SEI

Author

Julia H. Allen

This article was originally published in News at SEI on: January 1, 2005

What does it mean to govern for enterprise security or, stated differently, to govern an organization to achieve and sustain acceptable or adequate security? And why is the SEI's Networked Systems Survivability Program interested in this topic?

Our working definition of Governing for Enterprise Security is directing and controlling an organization to establish and sustain a culture of security in the organization's conduct (beliefs, behaviors, capabilities, and actions).[1]

Governing for Enterprise Security (GES) builds on and expands commonly described forms of governance. These include corporate governance, enterprise governance, and information technology (IT) governance.

Definitions of corporate governance typically include the relationships and incentives among boards of directors (or equivalent), senior executives, shareholders, and key stakeholders toward ensuring fiscal accountability, clear responsibility, and accurate reporting. Terms included in some definitions include probity (complete and confirmed integrity), due diligence, and standard of due care.

Corporate governance and enterprise governance overlap when the definition is expanded to include the "structure through which the objectives of the enterprise are set, and the means of attaining those objectives and monitoring performance are determined" [OECD 99, 04]. Structures and means may include, for example, policies (and their corresponding standards, procedures, and guidelines), strategic and operational plans, awareness and training, risk assessments, internal controls, and audits.

IT governance addresses the actions required to align IT with enterprise objectives and ensure that IT investment decisions and performance measures demonstrate the value

of IT toward meeting these.

While these definitions apply most often to commercial, for-profit corporations, they can also be interpreted and appropriately tailored for government, education, and non-profit institutions as well as organizations of any size.

Most senior executives and managers know what governance means and their responsibilities with respect to it. Our intent is to help them expand their governance perspective to include security, incorporating enterprise-wide security thinking into their organizations' day-to-day governance actions.

Motivation

The need to address security within organizations is growing in the public awareness. Customers are demanding it as concerns about privacy and identify theft rise. Business partners, suppliers, and vendors are starting to require it from one another, particularly when providing mutual network access. There is a wide range of current and pending U.S. national and international legislation that calls for organizations to exercise due diligence and demonstrate an acceptable standard of due care in how they manage their computing infrastructures and the information that such networks and systems create, transmit, and store, particularly when connected to the Internet. There are an ever-growing number of standards, guidelines, checklists, and assessment instruments with which organizations are expected to demonstrate some level of compliance. Certainly the U.S. federal government has recognized the potential impact of security breaches on critical infrastructures in its *National Strategy to Secure Cyberspace*, published in 2003, which contains a wide range of recommendations calling for improvement.

An organization's ability to mobilize to achieve and, more importantly, sustain a desired security state starts with executive sponsorship, enacted and sustained by governance. Those who lead, manage, set strategy, and are held accountable for an organization's success set the direction for how enterprise security is perceived, prioritized, managed, and implemented. If the responsibility for enterprise security is relegated to a role in the organization that lacks the authority, accountability, and resources to act and enforce, the enterprise security state will mirror this.

In many of the SEI's software engineering improvement initiatives, we find that executive awareness, understanding, and education are essential to achieve and sustain any level of improvement such that it becomes part of normal business conduct. To achieve widespread community improvement in security, we need to address this topic.

Coming Attractions

In a series of articles, we intend to examine some of the following elements of governance with respect to their role in governing for enterprise security. We will select those that have the greatest influence on achieving and sustaining an acceptable level of security (and what this means).

- Awareness and understanding —Governing boards and senior executives are aware of and understand the criticality of governing for enterprise security.
  - Protection of shareholder (or equivalent) value: They understand what actions are necessary to protect shareholder/stakeholder value with respect to enterprise security (such as protecting reputation and brand, and protecting customer privacy).
  - Customer satisfaction: They understand what enterprise security actions are necessary to retain current customers and attract new customers (such as sustained marketplace confidence in comparison to competitors).
- Strategies and plans —Strategies and plans for enterprise security demonstrate how they support business objectives.
  - Investments: Investments in enterprise security are aligned with and allocated so as to meet strategies and plans, taking risks into account (see risk management below). Costs are optimized.

- Reporting: Status against plans is regularly reported, up to the board. Performance against measures is monitored. Corrective action is taken when necessary.

- Policies —Policies, standards, guidelines, procedures, and measures for enterprise security exist and are regularly reviewed and enforced.
- Responsibilities —Responsibility and corresponding accountability and authority for enterprise security are clearly defined.
- Controls —Internal security controls are defined to effectively protect assets. Assets may include information, hardware, software, processes, services, physical facilities, knowledge, and people.
- Risk management —Risks to critical assets are identified and managed consistent with the enterprise's appetite and tolerance for risk. Asset protection investments are made commensurate with risk.
  - Liability—The enterprise understands its liability and exposure when connected to the Internet, and takes necessary due-diligence actions to minimize liability risk and exposure.
- Oversight—The enterprise is regularly evaluated and audited to ensure an acceptable level of compliance to requirements, both internal and external, such as regulations, standards, audit criteria, market sector requirements, and security requirements and objectives.
- Public disclosure —The enterprise is open to public disclosure of its security state, where such disclosure is required.

We intend to add to this list and welcome your feedback on its scope, content, and whether or not we are addressing concerns that are meaningful to your organization. Please send your remarks to Julia Allen at jha@cert.org.

Notes on definitions of governance

Corporate Governance

The Organization for Economic Development (OECD) defines corporate governance as follows:

"Corporate governance involves a set of relationships between a company's management, its board, its shareholders, and other stakeholders. Corporate governance also provides the structure through which the objectives of the company are set, and the means of attaining those objectives and monitoring performance are determined. Good corporate governance should provide proper incentives for the board and management to pursue objectives that are in the interests of the company and its shareholders and should facilitate effective monitoring." [OECD 99, 04]

Definitions of corporate governance are sometimes constrained to address financial reporting, the accountability of Boards of Directors (or equivalent), CEOs and other senior executives, and responsibilities to shareholders.

The recently published Corporate Governance Task Force Report Information Security Governance: A Call for Action [CGTF 04] defines corporate governance as "the set of policies and internal controls by which organizations, irrespective of size or form, are directed and managed."

EnterpriseGovernance

The Information Systems and Control Audit Association (ISACA) and the Information Technology Governance Institute (ITGI) define enterprise governance as follows. There is a degree of overlap between this definition and those for corporate governance:

"The set of responsibilities and practices exercised by the board and executive management with the goal of providing strategic direction, ensuring that objectives are achieved, ascertaining that risks are managed appropriately and verifying that the enterprise's resources are used responsibly." [ITGI 03]

This definition has also been adopted by The Chartered Institute of Management

Accountants (CIMA) [CIMA 04] and the International Federation of Accountants [IFAC 04].

IT Governance

Gartner states that

> "IT governance specifies the decision-making authority and accountability to encourage desirable behaviors in the use of IT. IT governance provides a framework in which the decisions made about IT issues are aligned with the overall business strategy and culture of the enterprise. Governance is about decision making per se —not about how the actions resulting from the decisions are executed. Governance is concerned with setting directions, establishing standards and principles, and prioritizing investments; management is concerned with execution." [Dallas 04]

ITGI defines IT governance as "the leadership, organizational structures, and processes that ensure that the enterprise's IT sustains and extends the enterprise's strategies and objectives." They additional state that "While governance developments have primarily been driven by the need for the transparency of enterprise risks and the protection of shareholder value, the pervasive use of technology has created a critical dependency on IT that calls for a specific focus on IT governance." [ITGI 03] Considering both of these definitions, much the same can be said for enterprise security and, in fact, ITGI has created a companion report on Information Security Governance [ITGI 01].

References

[CGTF 04]
Corporate Governance Task Force. "Information Security Governance: A Call to Action." National Cyber Security Partnership, April 2004. http://www.cyberpartnership.org.

[CIMA 04]
The Chartered Institute of Management Accountants. "Enterprise Governance —A CIMA Discussion Paper." 2004. See also [IFAC 04].

[Dallas 04]
Dallas, Susan, Bell, Michael. "The Need for IT Governance: Now More Than Ever (AV-21-4823)." Gartner, 20 January 2004.

[Hamaker 03a]
Hamaker, Stacey. "Spotlight on Governance." Information Systems Control Journal, 1, ISACA, 2003.

[Hamaker 03b]
Hamaker, Stacey; Hutton, Austin. "Principles of Governance." Information Systems Control Journal, 3, ISACA, 2003.

[Hamaker 04]
Hamaker, Stacey; Hutton, Austin. "Principles of IT Governance." Information Systems Control Journal, 2, ISACA, 2004.

[IFAC 04]
Professional Accountants in Business Committee; CIMA. "Enterprise Governance: Getting the Balance Right." International Federation of Accountants, February, 2004. See also [CIMA 04]. http://www.ifac.org.

[IIA 00]
The Institute of Internal Auditors et al. "Information Security Management and Assurance: A Call to Action for Corporate Governance." IIA, April 2000.

[IIA 01]
The Institute of Internal Auditors et al. "Information Security Governance: What

Directors Need to Know." IIA, 2001.

[ITGI 01]
Information Technology Governance Institute. "Information Security Governance: Guidance for Boards of Directors and Executive Management." Information Systems Audit and Control Foundation, 2001.

[ITGI 03]
Information Technology Governance Institute. "Board Briefing on IT Governance, 2nd Ed." ITGI, 2003.

[OECD 99]
Organisation for Economic Co-operation and Development. "OECD Principles of Corporate Governance." OECD, 1999.

[OECD 02]
Organisation for Economic Co-operation and Development. "OECD Guidelines for the Security of Information Systems and Networks: Towards a Culture of Security." OECD, 2002.

[OECD 04]
Organisation for Economic Co-operation and Development. "OECD Principles of Corporate Governance: 2004." OECD, 2004.

About the Author

Julia H. Allen is a senior member of the technical staff within the Networked Systems Survivability Program at the Software Engineering Institute (SEI), a unit of Carnegie Mellon University in Pittsburgh, PA. The CERT Coordination Center is also a part of this program.

Allen is engaged in developing and transitioning enterprise security frameworks and executive outreach programs in enterprise security and governance. Prior to this technical assignment, Allen served as acting Director of the SEI for an interim period of 6 months as well as Deputy Director/Chief Operating Officer for 3 years. Her degrees include a B. Sci. in Computer Science (University of Michigan) and an MS in Electrical Engineering (University of Southern California). She is the author of *The CERT Guide to System and Network Security Practices* (Addison-Wesley, June 2001).

1 The Organisation for Economic Co-operation and Development (OECD) discusses the need to develop a 'culture of security' in its *Guidelines for the Security of Information Systems and Networks* [OECD 02].

The views expressed in this article are the author's only and do not represent directly or imply any official position or view of the Software Engineering Institute or Carnegie Mellon University. This article is intended to stimulate further discussion about this topic.

Pas de Deux: Making the Two-Part Organization Work

NEWS AT SEI

Author

Paul C. Clements

This library item is related to the following area(s) of work:

Software Product Lines

This article was originally published in News at SEI on: January 1, 2005

In a 1947 speech to the House of Commons, Winston Churchill said, "Democracy is the worst form of Government, except all those other forms that have been tried from time to time." Sometimes the two-part organization for product lines seems to deserve that sentiment. In this column, we'll look at Cummins Engine Inc., a manufacturer of commercial diesel engines. Cummins has made the two-part organization work, but not without some hard lessons that are worth a closer look. One of the most common strategies is to have a separate core asset group to provide core assets to the product-builder groups, and this is the strategy that Cummins chose.

The two-part organization certainly has some advantages. A separate core asset group provides a focused setting where the product line as a singular entity can be managed and can evolve. It provides a central repository for the commonality and variability knowledge that is essential to running a product line. It is the first place where a new product idea can be explored for feasibility by seeing how many of the core assets the product would use. It's the place where cross-product commonality can be identified and exploited. It provides core assets that are generic across the entire family and not tilted toward any particular project.

But it can be tricky to run. For one thing, a product line manager has to staff a core asset group carefully. At one of our software product line workshops, a speaker rose to say that he could not entice any of his people to work in the core asset group because they felt it was too far removed from the real business--product delivery. But the next speaker said that in his shop, everybody wanted to work in the core asset group

Related Links

Training

Software Product Lines - eLearning

Software Product Lines

See more related courses »

because the software they wrote was used many times, and that's where they could have the biggest impact on their business. The core asset group generally requires the most talented designers and implementers, who are of course the most coveted by the project teams that will probably have to give them up.

Morale is a big issue with the two-part structure and is almost always overlooked. The core asset group forms a convenient target when things go wrong for the product builders, but they don't always seem to share in the warm spotlight of success when things go right. Members of the core asset group at Cummins told us of the time when a couple of errors were found in a delivered product, which is very bad news. E-mail flying around at the time accused the core assets of being responsible. It turned out not to be so, but somehow the retraction (if it appeared it all) was not circulated with the same enthusiasm or at the same shrill volume. And, they said a little wistfully, when things do go right, it's the product team that gets the tickets to the Indy 500 as a reward, not the core asset group. In thinking about organizational issues for product lines, Ron Temple, the Cummins vice-president in charge of the effort, lists team morale as an area of high concern:

> *A crucial challenge is maintaining team structure and morale. The "kudos" for product delivery will tend to go to the team closest to the end user customer in terms of customer and management visibility. The pressure and the "recriminations" for perceived delivery issues will tend to be aimed at the central team who are working to balance implementation, and therefore delivery, across multiple BUs. This tension can be ameliorated several ways – rotation of members between both types of teams, either on a temporary or long term basis, is one good answer. Living in your supplier's or customers' shoes for a while has very beneficial effects on your appreciation of the other person's challenges.*

Finally, the core group must be careful not to engender resentment by being seen to impose its own standards and processes and tools on the product groups. At Cummins, "core" was sometimes seen as a four-letter word, just like "evil" or "Borg" (as in "You will be assimilated; resistance is futile"). This can be frustrating for a group trying to juggle the concerns of multiple project teams and keep the big picture in everyone's view.

What made the two-part organization work at Cummins? In our view, four things: (1) having a champion in a place of authority; (2) the culture of cooperation within the company; (3) an effective funding model; and (4) achieving product-based consensus on priorities. We'll discuss the last two here.

The "Funding" practice area is concerned with who pays for the core assets. It may have seemed like an esoteric, possibly dry little practice area, but funding at Cummins is not only what gets the core assets built, it's what gets them used. Funding is used to drive the right behavior and establish a healthy organizational tension among the groups. Here's how it works: The budget for the core asset group is established. Each business unit is billed a portion of that budget in proportion to that business unit's overall sales. Thus, each business unit has every incentive to use the core assets because they're charged for them whether they do or not. Once again, the tendency to rely on cross-organizational sources is reinforced by policy. A funding model like this is a sure sign of whole-hearted endorsement from top-level management of the concept of core assets and therefore of product lines. It's more than lip service; management is voting with their pocketbooks. A by-product is that business units (or for that matter, individual projects) are strongly discouraged from bypassing the core asset group by building their own software. No project manager wants to stand up in front of his or her peers and admit to having the resources to commission duplicate work. That's a sign of a project that apparently has too much money, and that's a situation that can be quickly remedied.

This brings us to the fourth factor that makes the system work: There are weekly

technical planning meetings between the core asset groups and all of the product groups. These meetings are used to discuss new feature requests and trade information, and are invaluable at finding new areas of commonality. And it is at these meetings that the project managers put in their bids for what they need from the core asset group. They do this by prioritizing the change requests and problem reports that are in the core asset group's queue. The elegance of this approach is two-fold. First, it relieves the core asset group from having to decide what's important to the product groups, and thus removes them from being seen as legislating product priorities, something that would surely be resented. Second, it compels the product groups to work out among themselves what the actual priorities are. No project manager will stand before his or her peers and claim that all of the change requests are critical. The meetings force consensus among the product groups, and once again the good-of-the-whole culture is reinforced by organizational policy. The net effect is that it lets the core asset group be responsive to the product groups as a whole. The product groups regard them not as alien outsiders forcing assets on them, but as a service group driven to be as responsive as they can by the wishes of the group. "Having actual meetings is critical," one of the participants told us, "because you have to look the other managers in the eye when you say what your project's priorities are."

What if a product group really does need a core asset, but the core asset group is at capacity and the need isn't deemed a high priority by the group as a whole? Does the product group build it? No. If the need is really for something common, and the product group has the resources to build it, then those resources are re-allocated to the core asset group.

Cooperation among the core asset providers and the product builders has made the two-part organization succeed at Cummins; the Cummins people called it "concept alignment," the sense that all groups are working toward common goals for the common good. How they work is defined in the operational concept that must be communicated to all those involved.

About the Author

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

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

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

The views expressed in this article are the author's only and do not represent directly or imply any official position or view of the Software Engineering Institute or Carnegie Mellon University. This article is intended to stimulate further discussion about this topic.

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

Integrating Architecture Methods: The Case of Extreme Programming

NEWS AT SEI

Authors

Robert Nord

James E. Tomayko

This library item is related to the following area(s) of work:

Software Architecture

This article was originally published in News at SEI on: January 1, 2005

In a previous column (Rethinking the Software Life Cycle), we examined the traditional software development life cycle in the context of the architecture-centric methods that we have developed at the Carnegie Mellon Software Engineering Institute (SEI) over the past 10 years. These methods include the Architecture Tradeoff Analysis Method (ATAM) [Clements 02], the SEI Quality Attribute Workshop (QAW) [Barbacci 03], the SEI Attribute-Driven Design (ADD) Method [Bass 03], the SEI Cost Benefit Analysis Method (CBAM) [Bass 03], and SEI Active Reviews for Intermediate Design (ARID) [Clements 02]. In a subsequent column (Integrating Architecture Methods: The Case of the Rational Unified Process), we saw how these architecture-centric methods fit into the framework of plan-driven approaches as exemplified by the Rational Unified Process (RUP) [Kazman 04]. This column shows how these architecture-centric methods can be incorporated into agile approaches as exemplified by Extreme Programming (XP).

XP has a significant following. It got the name “extreme” from its tendency to “turn up the volume” of its practices. A cursory look at the practices shows that most are part of many methods, but XP ignores any other practice that does not appear on the list.

XP practices are based on four values: (1) communication, (2) simplicity, (3) feedback, and (4) courage:

1. Communication emphasizes person-to-person talking, rather than documents

Related Links

News

SATURN Conference Announces Additional Keynote, Conference Scholarships

Distinguished Speakers, Strong Technical Program Set for SATURN 2014

See more related news »

Training

Big Data - Architectures and Technologies

Documenting Software Architectures - eLearning

See more related courses »

that explain the software. Several XP practices call for an on-site customer, so a lot of time is spent communicating with that customer.

2. Simplicity is an XP value that calls for the solution of the customer's problem to be unpretentious. Both developers and customers easily understand the software solution.
3. Feedback means that everything done is evaluated with respect to how well it works. How well it works is indicated through the feedback gained from exercising every working part of the solution.
4. Courage means that developers are prepared to make important decisions that support XP practices while building and releasing something of value to the customer in each iteration. This could mean discarding code or taking the time to refactor the design when it turns out that previous decisions are inadequate.

In XP, the customer and the developers together determine the functionality that will be developed in a series of iterations. The first iteration influences the overall structure of the system. "The first iteration puts the architecture in place. Pick stories for the first iteration that will force you to create 'the whole system,' even if it is in skeletal form" [Beck 04]. Modifiability is implied by XP, but it is difficult to characterize. Developers develop the system incrementally, and when the system does not support new functionality, they refactor the design.

SEI architecture-centric methods can provide explicit and detailed guidance on eliciting the architectural requirements (such as modifiability), on designing the architecture, and on analyzing the resulting design.

In situations where requirements are changing rapidly and a lightweight approach is warranted, the concepts of quality attributes and architectural tactics can enhance the process of designing a system that will meet its requirements. Students participating in studio projects in the Master of Software Engineering Program at Carnegie Mellon University have been using the Architecture-Centric Development Method—that uses concepts from the QAW, the ADD method, and the ATAM—in this way.

When developing complex, large-scale applications, XP should be adapted to include more kinds of architectural information. Boehm and Turner demonstrate a solution for adapting XP to develop complex, large-scale applications by introducing elements of plan-driven methods [Boehm 04]. These elements include high-level architectural plans to provide essential big-picture information and use of design patterns and architectural solutions, rather than simple design, to handle foreseeable change.

The architecture developed by the ADD helps localize the effects of design changes caused by changing functional requirements, since the architecture is influenced by the quality attribute requirements and is not affected by changing functional requirements. Including architecture in this way might also delay refactoring. However, investing in the architecture means that it will take longer to get to code, because the first iteration is what some people call a "zero-feature release." In such a release, the architecture is put in place, but no user-visible features are delivered to the customer.

The benefit of including the SEI methods is to address quality attributes in an explicit, methodical, engineering-principled way. In summary

- The architecture-centric methods place an emphasis on quality attributes rather than functionality. They also help facilitate communication.
- The architecture-centric methods help fill gaps in the XP design process, by providing specific advice on
  - the elicitation and documentation of quality attribute requirements
  - which design operation will achieve a desired quality attribute response
  - how to analyze the result to understand and predict the consequences of the design decisions in terms of risks, tradeoffs, and ultimately return on investment (ROI)

- The architecture-centric methods all use common concepts: quality attributes, architectural tactics, and a "Views and Beyond" approach to documentation that leads to more efficient and synergistic use [Clements 03].

Table 1 shows how SEI concepts and methods can be used in keeping with the agile philosophy of rapid and flexible development and the XP values of communication, simplicity, feedback, and courage. More details are available in a technical note [Nord 04].

Table 1: The Architecture-Centric Methods and XP Values

XP Values	**Value Added Through Architecture-Centric Methods**
Communication	Cost-effective methods facilitate interaction among a diverse group of stakeholders. Stakeholders' concerns regarding quality attribute requirements are captured and communicated to developers so they influence design.
Simplicity	Architecture design is coarse-grained, and only enough architecting is done to ensure that the design will produce a system that will meet its quality attribute requirements. Architecture evaluation has a notion of triage and uses techniques such as utility trees and prioritization to focus efforts.
Feedback	Architecture evaluation provides early feedback for understanding the technical tradeoffs, risks, and ROI of architectural decisions. Risks are associated with technical decisions and business goals.
Courage	Risks are exposed early in the life cycle, giving developers justification for investing resources to mitigate them. Architecture allows for better planning so developers can better estimate the impact of requirements change. Change that is foreseen can be planned for and localized in the design.

References

[Barbacci 03]
Barbacci, M. R.; Ellison, R.; Lattanze, A. J.; Stafford, J. A.; Weinstock, C. B.; & Wood, W. G. *Quality Attribute Workshops (QAWs), Third Edition* (CMU/SEI-2003-TR-016). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003.

[Bass 03]
Bass, L.; Clements, P.; & Kazman, R. *Software Architecture in Practice*, *2nd edition*. Boston, MA: Addison-Wesley, 2003.

[Beck 04]
Beck, K. *Extreme Programming Explained: Embrace Change, Second Edition*. Boston, MA: Addison-Wesley, 2004.

[Boehm 04]
Boehm, B. & Turner, R. *Balancing Agility and Discipline: A Guide for the Perplexed*. Boston, MA: Addison-Wesley, 2004.

[Clements 02]
Clements, P.; Kazman, R.; & Klein, M. *Evaluating Software Architectures: Methods and Case Studies.* Boston, MA: Addison-Wesley, 2002.

[Clements 03]
Clements, P.; Bachmann, F.; Bass, L.; Garlan, D.; Ivers, J.; Little, R.; Nord, R.; & Stafford, J. *Documenting Software Architectures: Views and Beyond*. Boston, MA: Addison-Wesley, 2003.

[Kazman 04]
Kazman, R.; Kruchten, P.; Nord, R.L.; Tomayko, J.E. *Integrating Software Architecture-Centric Methods into the Rational Unified Process* (CMU/SEI-2004-TR-011), 2004.

[Nord 04]
Nord, R. L.; Tomayko, J. E.; Wojcik, R. *Integrating Software-Architecture-Centric Methods into Extreme Programming (XP)* (CMU/SEI-2004-TN-036), 2004.

About the Authors

Robert L. Nord is a senior member of the technical staff in the Product Line Systems Program at the Software Engineering Institute (SEI) where he works to develop and communicate effective methods and practices for software architecture. Prior to joining the SEI, he was a member of the software architecture program at Siemens, where he balanced research in software architecture with work in designing and evaluating large-scale systems. He earned a Ph.D. in Computer Science from Carnegie Mellon University. Dr. Nord lectures on architecture-centric approaches. He is co-author of Applied Software Architecture and Documenting Software Architectures: Views and Beyond.

James E. Tomayko Dr. James E. Tomayko is a teaching professor at the School of Computer Science at Carnegie Mellon and a part-time senior member of the technical staff of the Software Engineering Institute (SEI). He is the director emeritus of the Master Software Engineering Program in SCS.

Previously he was leader of the Academic Education Project at SEI. Prior to that, he founded the software engineering graduate program at the Wichita State University. He has worked in industry through employee, contract or consulting relationships with NCR, NASA, Boeing Defense and Space Group, CarnegieWorks, Xerox, the Westinghouse Energy Center, Keithley Instruments and Mycro-Tek. He has given seminars and lectures on software fault tolerance, software development management, fly-by-wire, and software process improvement in the United States, Canada, Mexico, Argentina, Spain, Great Britain, South Africa, Germany, China, and Columbia. Tomayko's courses on managing software development and overviews of software engineering are among the most widely distributed courses in the SEI Academic Series.

Tomayko has had a parallel career in the history of technology, specializing in the history of computing in aerospace, and has written five books and several articles on spacecraft computer systems and software, concentrating primarily on NASA's systems. Dr. Tomayko is on the editorial staff of the *IEEE Annals of the History of Computing*.

The views expressed in this article are the author's only and do not represent directly or imply any official position or view of the Software Engineering Institute or Carnegie Mellon University. This article is intended to stimulate further discussion about this topic.

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

Nine Characteristics of a COTS and Reuse Management Plan

NEWS AT SEI

Authors

Bill Anderson

Edwin J. Morris

Dennis B. Smith

Mary C. Ward

This library item is related to the following area(s) of work:

System of Systems

This article was originally published in News at SEI on: January 1, 2005

Commercial, military, and other government organizations continue to increase their reliance on reused software to provide major capabilities in new systems. This reused software goes by many different labels, including commercial off the shelf (COTS), government off the shelf (GOTS), shareware, freeware, open source, and non-developmental items.

When components are used for large-scale systems of systems, they can have unforeseen effects on other parts of the systems. One way to minimize these effects is to develop a COTS and reuse management plan, similar to one that we recently developed in collaboration with a large DoD program.

In developing this plan, we identified nine characteristics that are fundamental to long-term use of complex reusable software. These characteristics include

1) A product line strategy: A software product line is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way. Building a set of software systems as a software

Related Links

Training

Migrating Legacy Systems to SOA Environments - eLearning

Service-Oriented Architecture: Best Practices for Successful Adoption

See more related courses »

product line, compared to developing the systems one at a time in isolation from each other, has been shown to shorten development time, increase productivity, increase quality, and reduce cost.

A product line strategy provides for systematic reuse. This contrasts with a “clone-and-own” approach in which a reusable asset is discovered, modified, and installed in the product. When using such an approach, maintenance and evolution are no longer shared with other members of the product family.

To effectively implement a software product line, the reuse management plan should define processes for

- determining when a product line is appropriate
- developing a flexible product line architecture
- identifying the core components within the product line
- building a production plan that describes how the core components will be reused
- defining a construction management process to control maintenance and evolution of the product line

2) An iterative process: Use of COTS components does not lend itself to waterfall-type development, since neither reused components nor our understanding of them is static. Managing the dynamic nature of components and their interactions with each other and the rest of the system is a key to effective reuse.

A COTS and reuse management plan should follow an iterative approach to development and long-term sustainment of systems that supports

- iterative refinement of system requirements, architectures, and reuse-component commitments to balance the tension between users and implementers and reuse components effectively
- early identification of risks and application of risk-mitigation strategies

3) A reuse component manager: It is also important to consolidate management activities under a central authority. This authority serves as the clearinghouse for information and the organizer of reviews and other tasks. Each component considered for reuse should be assigned a reuse component manager.

Among the responsibilities of the reuse component manager are

- notifying affected organizations of plans or changes to plans for use of a component
- organizing and stewarding life-cycle activities such as component (re)evaluations, version upgrades, analysis of patches, reviews
- monitoring risks associated with use of the component
- directing market-watch activities for the component
- developing and implementing a strategy to create and manage vendor/provider relationships

4) Risk-based management of components: Virtually any characteristic of a component or component provider may increase the risk for a specific system. The processes that may require tailoring based on reuse component risks are

- requirements processes to focus on putting critical components to best use
- evaluation processes that change the scope and extent of evaluation activities
- review processes that assure that high-risk components are reviewed with sufficient frequency by the right people
- engineering processes to focus proper attention on the component and its interactions with the rest of the system
- problem and risk reporting processes
- processes for validating and approving patches and upgrades

5) Full life-cycle coverage: Our experience suggests that long-term costs of maintaining reused code often exceed initial procurement costs. Life-cycle challenges include

- guidance on architecting, designing, integrating, and testing when using reused components
- problem reporting and management
- impact analysis for requirements changes, new versions, problems, and patches
- license management
- managing relationships with component providers
- metrics for reuse components
- periodic reviews of the health of the component (health check)

6) Aggressive evaluation and selection of components: A wide-ranging and aggressive evaluation and selection process for components should be documented as part of the plan.

This process includes

- a make-reuse decision that is based on analysis of the expectations for the component within the system and of the marketplace of components that can be reused
- organized planning for evaluation and selection of components
- evaluation criteria for critical aspects of the component
- defined processes for evaluation and selection of components (Bergey, J.; O'Brien, L.; & Smith, D. Options Analysis for Reengineering (OAR): A Method for Mining Legacy Assets (CMU/SEI-2001-TN-013, ADA395201). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2001.) that provide guidance on evaluation and selection of large-scale assets (like subsystems) and COTS components, respectively; these processes can be modified to fit other types of reuse components.
- mechanisms for capturing not only information about components but also impact of components on requirements, architecture, design, implementation, testing, and deployment of the system

7) A complete historical record: Central to any good reuse management plan is complete documentation about reuse components considered, selected, and used within the system.

Data should include

- history of evaluation/selection of reuse components, including criteria, rationale for criteria, data/results of evaluation, and rationale for selection
- licensing information
- history and archive for version releases
- metrics data and analysis
- identification of other components that depend on the reuse component and components on which the reuse component depends
- results of periodic reviews and component health checks
- information about preparation of a component version for inclusion in the system (e.g., tailoring, modification, parameters and settings)

8) A component health check: The status of reuse components should be reviewed frequently (for instance, biannually) to determine whether existing strategies for use of the component remain valid.

The component health checkup normally considers four primary sources of information:

1. information gathered by the component manager while performing his/her activities
2. information summarized from tracking the reuse component (e.g., problems, risks, patches, versions, vendor plans)
3. information gathered during market-watch activities
4. information from users about evolving expectations for the component

9) Metrics for improvement: The typical metric associated with reuse components is an equivalent SLOC count that is intended to represent the effective savings of procuring rather than building the component. Often this count includes measures of the integration effort associated with the component.

Later in the development process, actual performance in incorporating the reuse component against earlier estimates is often tracked. In addition other sources of data can include

- stability or instability of requirements related to reuse components
- a summary and analysis of defects in reuse components
- stability of contacts and consistency of information regarding the reuse components
- frequency of release schedule and success in meeting schedules
- complexity of adaptation code

About the Authors

Edwin Morris is a Senior Member of the Technical Staff at the Software Engineering Institute, assigned to the Integration of Software-Intensive Systems (ISIS) Initiative. He is currently investigating approaches to achieving technical interoperability between complex systems and programmatic interoperability between the organizations that build and maintain them. Previous activities involved improving processes and techniques for the evaluation and selection of COTS products, and the development of the COTS Usage Risk Evaluation (CURE) technology. Before coming to the SEI, Ed developed custom operating systems for embedded microprocessors along with support tools to predict and monitor the performance of real time systems.

William B. Anderson is a senior member of the SEI technical staff. Bill's research interests include integration and interoperability of complex software systems, COTS and reuse management, cost estimation, and business case justification of complex systems. A former Vice President for a Fortune 500 company, Bill is broadly experienced with factory floor and business; processes, support systems, automation, and management. He has many years of experience in large system project management and has successfully led operational, financial, product line, and new product launch groups.

Dennis Smith is the Lead for the SEI Initiative on the Integration of Software Intensive Systems. This initiative focuses on addressing issues of interoperability and integration in large scale systems and systems of systems. Earlier, he was the technical lead in the effort for migrating legacy systems to product lines. In this role he developed the method "Options Analysis for Reengineering, OARS to support reuse decision-making. Dr. Smith has also been the project leader for the CASE environments project. This project examined the underlying issues of CASE integration, process support for environments and the adoption of technology.

Dr. Smith has published a wide variety of articles and technical reports, and has given talks and keynotes at a number of conferences and workshops. He has an M.A. and PhD from Princeton University, and a B.A from Columbia University.

Share This Page

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

A Personal Quality Strategy

NEWS AT SEI

Author

Watts S. Humphrey

This library item is related to the following area(s) of work:

Process Improvement

This article was originally published in News at SEI on: January 1, 2005

This is the fourth column in a four-column series on software quality and security. The first column discussed the software quality problem, why customers don't care about quality, and how bad software quality really is. The second column described why testing alone cannot produce much higher quality software than we get today. In the third column, I described the quality attitude and how software professionals and their managers must change their view of quality if we are to make much headway improving software quality and security. This column discusses the stages of quality and strategies for addressing the quality problems at each stage. It also describes practices to consider as you improve your personal performance as a software developer. Finally, I comment on the challenges ahead and how the strategies described here can help you to address them.

Quality Stages

There are many ways to look at quality, but, from a software development perspective, we can think about quality as having five stages.

Stage 1 – Basic code quality: syntax and coding constructs

Stage 2 – Detailed design: the logical construction of programs and the actions required so that these programs perform their specified functions

Stage 3 – High-level design: system issues such as interfaces, compatibility, performance, security, and safety

Stage 4 – Requirements focused: determining the meaning of the requirements and

Related Links

News

SEI to Co-Sponsor 26th Annual IEEE Software Technology Conference

SEI Cosponsors Agile for Government Summit

Training

See more related courses »

Events

Team Software Process (TSP) Symposium 2014

Nov 3 - 6

particularly deducing what is written between the lines.

Stage 5 – User driven: users and what we must do to provide them with truly great products. While last in this list, user concerns must have top priority.

Stage 1 – Basic Code Quality

At the stage of basic code quality, quality is personal. Either you are fluent in the programming language or you are not. If you are fluent, your principal concerns are typos and occasional obvious mistakes. I once tracked my errors for a large data-entry job and found that I made 1.74 errors per thousand keystrokes, or 2.4 errors per hour of key entry. Since a line of my C++ code averages 17 keystrokes, my key-entry error rate alone injected 29.5 defects per 1,000 lines of code.

This was just typing. I did not consider program semantics, functions, or design. To correct this type of basic mistake, you need to track the defects and understand the errors that caused them. Until you do, you cannot prevent these types of defects or get better at finding and fixing them. Many developers object to tracking defects that the compiler could quickly find. The reason to record these defects, however, is to understand the mistakes you make no matter how they are found. Once you have defect data, analyze it. By recording your defects and analyzing the data, you are likely to cut your defect-injection rate by about 50%. Also, keep recording the defect data. If you don't, your injection rate will creep back up.

If you are not yet fluent with the programming language, take the same three steps: track your defects, analyze the data, and continue doing this as a regular part of your programming work. By tracking and analyzing my defects and using the defect data to guide my personal design and code reviews, I substantially cut the time it took me to learn a new programming language and also dramatically improved the quality of my products.

The obvious next question is, "Why bother with these Stage 1 defects, since the compiler will find most of them anyway?" The brief answer is that tools and testing will not find a significant number of your defects, and any that you miss will be very expensive for you, the testers, or the users to find and fix later. (For a more complete answer, consult my previous three columns mentioned above.) If you don't clean up the Stage 1 defects first, they will make it harder for you and your teammates to find and fix the more sophisticated defects at the higher quality stages. You will also waste a lot of test time fixing defects that you could have quickly found and fixed beforehand.

Stage 2 – Detailed-Design Quality

At the detailed-design stage, the problems are more sophisticated. Most programmers make design mistakes not because they don't know how to design, but because they never actually produced a design. They may have drawn some bubble charts or sketched a few use cases, but they never reduced these designs to a specification for writing code.

The practice of designing while coding is error prone. From data on 3,240 programs written in Personal Software Process (PSP) courses, the SEI has found that experienced developers inject fewer defects when designing (2.0 defects per hour) than when they design while coding (4.6 defects per hour). If you want low-defect designs, you must produce those designs, instead of just creating them while coding.

There are two big advantages to producing designs. First, you will make less than half as many design mistakes, and second, you will have a design that you can review and correct. However, in doing the design review, use good review methods. This is too big a subject to cover here, but it is covered in my new book *The Personal Software Process – A Discipline for Software Engineers* to be published in March 2005.

To do design reviews properly, you must spend enough time doing them. From the data on the same 3,240 PSP programs, the average defect-removal rate during design reviews was 3.3 defects per hour. Therefore, if you inject defects at the rate of 2.0 per hour and remove them at 3.3 per hour, you must spend about 36 minutes reviewing

the design for every hour you spent producing it.

The detailed-design stage is particularly important because this is where you design the logic paths that must be tested. If some of the paths through your module have subtle defects and if you don't find and fix them, they will be left for the testers or users to find. Since you presumably will have tested your program before passing it on to test, the remaining defects will be the ones that did not show up in your testing. In other words, these defects do not prevent the program from working except under specialized conditions. To find them, the test must cover the right paths and it must have the proper parameter or variable values.

Assuming that your programs have about the same proportion of branch instructions as my C++ programs, your typical 500 line-of-code (LOC) module would have about 50 branch instructions. From the data in Table 1, your program would then have about 3,400 possible test paths, any one of which could contain a design defect that you missed. (For more on the testing maze, see my previous column in news@sei.) That is why the strategy of testing in quality takes so long and produces defective products.

Table 1. Testing Paths

Maze Size	Branches	Paths	Maze Size	Branches	Paths
1	1	2	11	121	705,432
2	4	6	12	144	2,704,156
3	9	20	13	169	10,400,600
4	16	70	14	196	40,116,600
5	25	252	15	225	1.55E+08
6	36	924	16	256	6.01E+08
7	49	3,432	17	289	2.33E+09
8	64	12,870	18	324	9.08E+09
9	81	48,620	19	361	3.53E+10
10	100	184,756	20	400	1.38E+11

The key practices at Stage 2, the detailed-design stage, are to produce complete designs, review the designs yourself, and have your teammates carefully inspect them. Then, of course, implement the program and thoroughly review, inspect, and test it to make sure that the code properly reflects the design. If you and your teammates do this for every module you and your team develop, you will improve the quality of your programs by at least 10 times and probably much more. You will also save a lot of test time.

Stage 3 – High-Level Design Quality

Through Stage 2, the defects are yours. You inject them, and you are the best person to find and fix them. Above this stage, you must work with others. The defects at Stage 3 concern the interfaces, interdependencies, and interactions of your program with the other parts of the system. Defects in any module could affect system properties such as

performance, security, and safety. These properties result from the correct operation of all or most of the parts of the system. This is another case where sound design practices are important. For security, for example, a properly-produced high-level design would specify the authentication practices and the data-security and protection conventions that the modules should follow.

At Stage 3, the key quality practices are first, to get and review the high-level design specifications for your module. If these specifications don't exist or are inadequate, work with your teammates and system designers to clarify the design specifications. Second, after you obtain these specifications, follow them in producing the module design. Then, third, review the design to ensure that it meets these specifications and that it will work with all of the other modules. If you don't do this, there is a good chance that the modules you develop will not work properly with the rest of the system. Often, such problems cannot be fixed without a major module redesign or a complete module replacement. Finally, get your team's help in thoroughly inspecting and correcting the design. This is particularly important because, at this stage, your products begin to address system-level issues that you may not even be aware of.

Stage 4 – Requirements-Focused Quality

At Stages 1, 2, and 3, you work with familiar issues. At the requirements stage, however, you are no longer an expert. The two principal difficulties with requirements problems are, first, that requirements are not your field of specialty. Therefore, it is easy to think that you understand something when you do not. Second, misunderstandings and errors in requirements interpretation can easily cause you to build the wrong product.

If not caught early, requirements misunderstandings can be fatal. This is why you should start with a thorough examination of the requirements and resolve any confusion or uncertainty with requirements experts before starting to design the product. While many requirements details need not be settled at this point, it is hard to know what is a detail and what is fundamental. Once you are familiar with the requirements, settle the critical points before starting the design work and resolve the remaining issues in parallel with the design and implementation work. The two key practices at Stage 4 are to understand the requirements and to resolve any confusion or uncertainty before starting on the design. In implementing these practices, get your team's help as well as the help of the organization's systems designers or requirements experts. Some of these people have been thinking about this requirement for a long time and will likely understand the user's needs better than you possibly could with just a few brief weeks or months of exposure.

Stage 5 – User-Driven Quality

The user stage is a totally different ballgame. Here, as shown in Figure 1, the problems are not with what you know or even what you don't know, they are with what you don't know that you don't know. This problem is particularly tricky because the users often don't understand the issues either. A truly great product must perform a desirable user function in a convenient and elegant way. While the users may think that they know what they need and have strong opinions on how the product should work, they will often be wrong. They won't completely understand or be able to specify the functions that they want, and their view of how to build a product to perform these functions will approximate what they do today, rather than some elegant new approach.

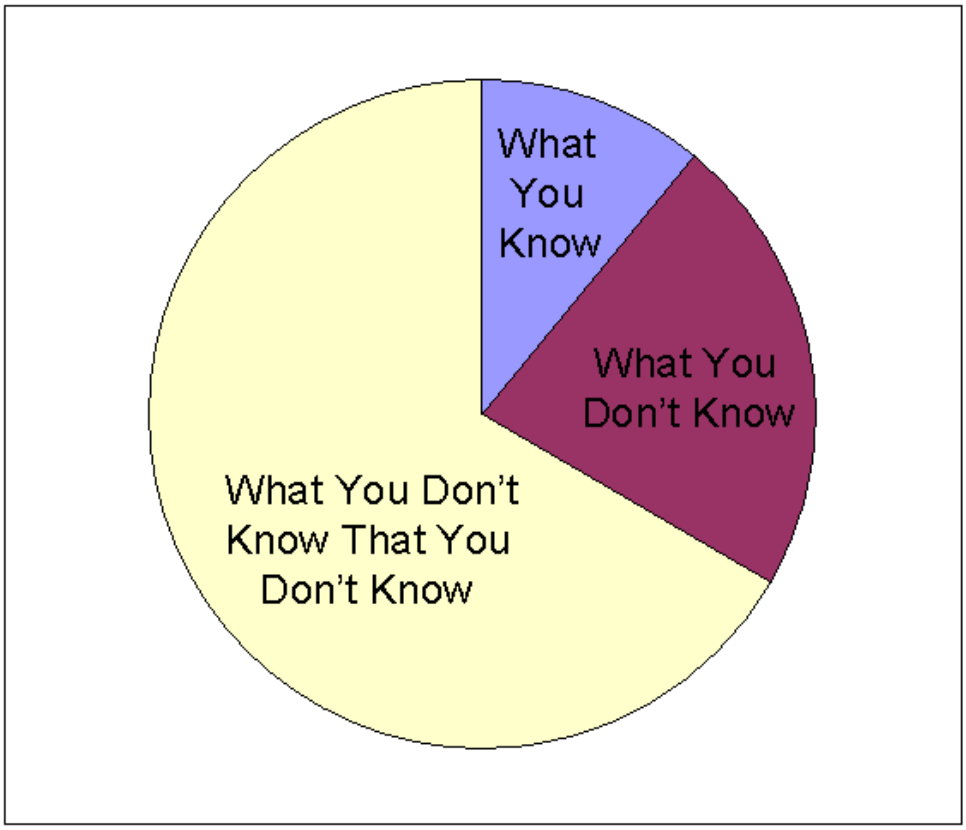

Figure 1. What We Know and Don't Know

The challenge at the user stage is to get a clear understanding of what the users think they want and then to develop an intuitive sense for what they really need. Once you do this, keep thinking about the problem and you could come up with a truly creative solution. This is how breakthroughs like the spreadsheet or mouse were conceived. A developer who understood the problem had a "crazy" idea that worked. While these creative leaps don't happen often, preparing for them will help you to build a fine product. You will have solved a problem in a better way than you otherwise would have, and you will have given yourself the chance to do something great.

What makes the user-driven stage so different is that elegant solutions often change the problem, and sometimes they change it in fundamental ways. Your objective at this stage should be to develop a deep enough intuitive understanding of the user's needs so you can see the opportunities for dramatic departures that will transform the problem while enabling a breakthrough solution.

Putting It All Together

Quality is an individual issue. If you really want to do great work, there are lots of ways to do it. You may have to settle for incomplete requirements and specifications, and you may not even be able to talk to any users, but you can always do a first-class job at Stages 1, 2, and 3. Keep thinking about the higher quality stages and what you can do to extend yourself and your team to do great work. You may not succeed often, but it is worth the try. Major advances take time, a lot of insight, and some perspiration. They also take an occasional inspiration. Even if it takes many years, a great product is something that you will always be proud of. So keep trying to do quality work. It will make your life much more rewarding.

Acknowledgements

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

In closing, an invitation to readers

In these columns, I discuss software issues and the impact of quality and process on

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

Thanks for your attention and please stay tuned in.

Watts S. Humphrey
watts@sei.cmu.edu

About the Author

Watts S. Humphrey founded the Software Process Program at the SEI. He is a fellow of the institute and is a research scientist on its staff. From 1959 to 1986, he was associated with IBM Corporation, where he was director of programming quality and process. His publications include many technical papers and several books. His most recent books are *Introduction to the Team Software Process* (2000) and *Winning With Software: An Executive Strategy* (2002). He holds five U.S. patents. He is a member of the Association for Computing Machinery, a fellow of the Institute for Electrical and Electronics Engineers, and a past member of the Malcolm Baldrige National Quality Award Board of Examiners. He holds a BS in physics from the University of Chicago, an MS in physics from the Illinois Institute of Technology, and an MBA from the University of Chicago.

The views expressed in this article are the author's only and do not represent directly or imply any official position or view of the Software Engineering Institute or Carnegie Mellon University. This article is intended to stimulate further discussion about this topic.

Software Acquisition Survival Skills: Helping the DoD and Government Program Offices Improve Acquisition of Software and Systems

NEWS AT SEI

Author

Joseph P. Elm

This library item is related to the following area(s) of work:

Acquisition Support

This article was originally published in News at SEI on: February 1, 2005

The SEI's Acquisition Support Program (ASP) assists the U.S. Department of Defense and civil agency program offices responsible for acquisition of software and systems. To accomplish this, the SEI has developed a three-point approach.

1. Needs Analysis: Understand and characterize the acquisition environment by gathering and analyzing issues and problems associated with advancing the state of the practice for acquiring software-intensive systems.
2. Acquisition Improvement: Work directly with key acquisition programs to help them achieve their objectives. Apply new technologies by conducting experiments with maturing SEI products and services in real-world acquirer contexts. Establish a delivery capability to meet the strategic software acquisition objectives of the DoD and civil agencies. As needed, provide an on-site presence to assist acquisition officials in the improvement of their software-intensive system acquisition activities.
3. Knowledge Integration and Transfer: Capture knowledge from engagements with acquisition organizations, integrate it with lessons learned from other similar work, and help transfer that knowledge for the betterment of the acquisition community. The outputs of these activities may include conferences, workshops, courses, briefings, technical reports, articles, advocacy, and participation in acquisition communities of practice.

Related Links

News

SEI Cosponsors Agile for Government Summit

Training

Data Rights and DoD Acquisition

SGMM Navigator Training

See more related courses »

(< 5 minute) survey.

Software Acquisition Survival Skills

As part of knowledge transfer, the SEI has developed "Software Acquisition Survival Skills" (SASS), a three-day training course aimed specifically at acquisition professionals tasked with acquiring software or software-intensive systems. This course provides an overview of skills needed by the program manager and program office staff to successfully acquire systems and software.

SASS was developed by first researching the problems that typically beset acquisition offices. SEI staff members studied a number of published reports addressing acquisition by the DoD and other civil agencies, and examined the training provided by the DoD to acquisition professionals, and extracted the major issues. (We reviewed the DAU acquisition courses, earlier DoD training [ARES, Boldstroke], and training from other sources [Aerospace, NRO].) We surveyed program managers from the DoD and other civil agencies, staff from the Office of the Secretary of Defense, and SEI staff involved with acquisition programs, asking them "What are the five biggest problems that you see in programs?" We categorized and prioritized the responses based on the frequency of occurrence and severity of impact as reported by respondents. We started at the top of the list and built instructional modules for each category. Because the course duration was specified at three days, we had enough time for eight modules. The top eight problem areas (listed below in no particular order) became the topics for the SASS instructional modules.

- Risk Management
- Process Management
- Requirements Development and Management
- Pre-Award Activities of the Program Office
- Systems Engineering
- Technical Evaluations of Work Products
- Software Architecture
- Program Management Using Metrics

Each topic is presented from the perspective of the acquirer and examines the challenges posed in acquisition programs. For example, the Process Management instruction module covers the role of the program manager in defining and managing program office and program processes, in overseeing contractor process management, and in integrating the process sets of the program stakeholders (e.g., program office, contractors, and subcontractors). The module also presents steps the Program Manager can take to improve Program Office processes, and to encourage contractors to improve their processes.

While the breadth of the information covered and the short duration (three days) of the course precludes in-depth coverage of these topics, instructors do attempt to provide actionable recommendations for the students. In addition to background information for each topic, SASS provides

- Warning signs to look for in programs (e.g., *Warning sign of problems in Risk Management*: An examination of the risk database reveals that no updates have been performed in several months.)
- Strategies to prevent the occurrence of problems in programs (e.g., *Strategy to prevent problems with metrics in program management*: Train your program office staff on the development and use of program metrics. Use these metrics as "early warning" indicators of program problems.)
- Strategies to recover from problems in programs (e.g., *Recovering from problems stemming from architecture suitability*:
    - Evaluate your architecture's conformance to the defined quality attributes, functional requirements, and system constraints.
    - Identify and prioritize gaps (involve your stakeholders).
    - Spend the time and money to develop a better architecture for

comparison.

- Prioritize your changes based upon impact upon the project and the importance of conformance.)

To help students apply the concepts presented, each instructional module ends with a series of "What Do I Do Now?" slides, providing clear suggestions for next steps that can be taken at various stages of the program. Additionally, instructors provide a number of checklists, tools, and references that the students can use in the execution of their programs.

For students needing more detailed information on the topics presented in the course, the SEI offers a number of companion courses covering specific topics in more detail:

- Continuous Risk Management
- COTS-Based Systems for Program Managers
- COTS Software Product Evaluation for Practitioners
- Defining Software Processes
- Introduction to CMMI
- Managing Software Projects with Metrics
- Mastering Process Improvement
- Process Improvement Overview
- Documenting Software Architectures
- Software Architecture: Principles and Practices

The SASS is a relatively new course, offered for the first time in 2004. Over the past year, SASS has been offered 15 times, and more than 200 students have been trained.

It has been offered as a public course five times (three times at the SEI in Pittsburgh and twice at the SEI in Arlington), with attendance from Army, Navy, Air Force, civil agencies, and commercial contractors. Public offerings in 2005 are scheduled for

- *SEI-Arlington*
  June 28, 29, 30
  December 14, 15, 16
- *SEI-Pittsburgh*
  September 27, 28, 29

Visit our website to register for the public course.

SASS has also been offered 10 times on-site at Army and Air Force program offices. On-site offerings have the greatest impact because the instructors can address a broad cross-section of the staff at a single program office and, in addition to presenting the course materials, discuss the specific issues affecting that program. Comments from program office staff, as well as follow-up contact with these program offices have shown the benefits achieved as the programs adopt the concepts presented in the course. Visit the training section of our website to learn how you can bring the SASS to your site.

info@sei.cmu.edu
412-268-5800

SEI Celebrates 20 Years of Software Engineering Excellence

NEWS AT SEI

This article was originally published in News at SEI on: February 1, 2005

The year was 1984. President Ronald Reagan was elected to his second term in office. The Summer Olympics were held in Los Angeles with tennis great Steffi Graf winning a gold medal and the U.S. being introduced to gymnastics phenomenon Mary Lou Retton. *The Cosby Show* and *Miami Vice* were on television, and the first Indiana Jones movie was in theaters.

1984 also marked dramatic and profound advances in information technology. Apple introduced its Macintosh computer; there were roughly 1,000 users on the Internet; Dr. Paul Mockapetris and Jonathan Postel were working on the domain name system (DNS), and the U.S. military portion of the ARPAnet was broken off as a separate network, the MILNET.

And in December 1984, the U.S. Department of Defense (DoD) awarded a contract to Carnegie Mellon University to establish the Software Engineering Institute (SEI). The purpose of this institute was to study software as a technology and software engineering as an emerging discipline.

Two years earlier, the DoD launched a software initiative with three deliverables, one of which was a Software Engineering Institute. The contract for the SEI was opened to competitive bidding—a first for a federally funded research and development center—and 40 institutions submitted proposals. Led by Dr. Angel Jordan, then Carnegie Mellon provost and later SEI director, the university drew on its strength and reputation in computer science and its long-term vision for the institute. Carnegie Mellon also distinguished itself from the other competitors in that it intended to put all of the SEI's operations on one campus.

The SEI's first office, located in the Pittsburgh neighborhood of Shadyside, opened its doors in January 1985 and originally employed 55 people, including 12 managerial staff and 19 researchers. Experts from government, academia, and industry brought significant technical capabilities and experience to the SEI's research team.

In 1987, employees moved into the current headquarters on Fifth Avenue in Oakland. At its opening celebration, former Carnegie Mellon President Dick Cyert said at the

time that “the aim of the institute is to find the techniques to increase the productivity of software programmers. It is in the supply and quality of software programmers that our society is deficient. We have not fully appreciated their importance...I believe we do now, and that this institute symbolizes this effort.”

The SEI has grown significantly in its 20 years. The institute continues to hire the best of the best in the field and has become the preeminent institute in software engineering. The SEI currently has more than 500 employees with office locations in Pittsburgh, Washington D.C., Huntsville, AL, and Frankfurt, Germany.

Since 1996, the SEI has been organized into technical initiatives in process management, software-intensive systems, and network security. The SEI is best known for two major innovations: the development of the Capability Maturity Modeling framework for software process improvement and the establishment of the CERT Coordination Center (CERT/CC) in 1988.

The SEI has also influenced the design and acquisition of software-intensive systems. In one significant area of SEI research, the SEI has identified how organizations can reuse the labors of software development work in the form of software product lines and how decisions about software architecture affect the long-term viability of a product or system. The SEI is also identifying and maturing software engineering methods and techniques for broad-based and sustained integration and interoperability across components, systems, and systems of systems. Through the SEI's Acquisition Support Program, the SEI is also helping the DoD and other government acquirers improve the acquisition of software-intensive systems.

Celebrating Today and the Future

As the SEI celebrates its 20th anniversary, the SEI will honor those who established the SEI, those who have provided guidance and leadership, and long-time employees.

In January, distinguished guests from Carnegie Mellon, Pennsylvania congressional leaders, current and former members of the SEI Board of Visitors, and SEI past directors joined SEI Director and CEO Paul D. Nielsen at the Renaissance Hotel in Pittsburgh to commemorate the SEI’s 20th anniversary. Keynote speakers included Carnegie Mellon President Jared Cohon, Carnegie Mellon Provost Mark Kamlet, SEI Chief Operating Officer Clyde Chittister, and past Directors Larry Druffel and Julia Allen.

Nielsen attributes the SEI’s far-reaching influence to strong leadership and support

(< 5 minute) survey.

from the DoD, the Board of Visitors, and Carnegie Mellon leadership. Cohon told the guests that “the SEI has partnered with many organizations and enhanced the name of Carnegie Mellon worldwide. The SEI is a world-class institute. We [Carnegie Mellon] are very proud of the SEI and its commitment to advancing the state of the practice of software engineering.”

Nielsen stated that while the SEI still has a Department of Defense focus, the SEI’s influence has spread throughout the world. “Today, software drives economic growth and competitiveness. It plays a crucial role in a nation’s security,” Nielsen said. “The SEI is helping improve the state of the practice of software and systems engineering. The SEI is truly an outstanding organization, and I feel very fortunate to be part of it as we begin the next 20 years of success.”

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

Model-Driven Architecture: Moving from Concept to Practice

NEWS AT SEI

Author

Grace Lewis

This library item is related to the following area(s) of work:

Software Architecture

System of Systems

This article was originally published in News at SEI on: February 1, 2005

The SEI is examining technologies and approaches for the construction of systems that are required to interoperate with other systems, with the purpose of examining the gaps between what these technologies and approaches offer and what users expect of them. One of these approaches is model-driven architecture (MDA), produced and maintained by the Object Management Group (OMG), an open membership, not-for-profit consortium that produces and maintains computer industry specifications for interoperable enterprise applications.

The goal of MDA is probably one you have heard before: to separate business and application logic from its underlying execution platform technology so that (1) changes in the underlying platform do not affect existing applications, and (2) business logic can evolve independently from the underlying technology [Lewis 04, OMG 03]. A tool that implements the MDA concept allows developers to produce models of the application and business logic and, by means of a model-to-code translator, generate code for a target platform.

At this point you might be asking yourself: how is this different from CASE environments, automated code generators, or even Java? There is something different and interesting about MDA and it is that OMG is working on specifications such as Unified Modeling Language (UML), XML Metadata Interchange (XMI), Meta-Object Facility (MOF), and Common Warehouse Meta-model (CWM) to provide a common modeling representation for vendors to use in their tools. By using this common

Related Links

News

SATURN Conference Announces Additional Keynote, Conference Scholarships

Distinguished Speakers, Strong Technical Program Set for SATURN 2014

See more related news »

Training

Big Data - Architectures and Technologies

Documenting Software Architectures - eLearning

See more related courses »

modeling representation to build applications, not only would applications have better chances of achieving interoperability with other applications, but models could be shared between tools, and therefore the same application could be deployed on multiple platforms. Unfortunately, most current MDA tools have focused on the automated code generation aspects and not on the real strength of MDA, which is the common modeling representation.

Nonetheless, many people are becoming attracted to MDA because of other attributed benefits, such as reduced life-cycle cost, reduced development time for new applications, improved application quality, and a more rapid response to changes in technology, even though the data that supports these claims is limited. To better separate fact from fiction, the SEI has set up an experimentation environment to test these claims. What follows are some preliminary findings related to reduced development time for new applications.

Reducing Development Time with MDA

One element that factors into development time is the amount of code that has to be written that is not generated by the tool. There is currently a wide variation in the amount of code that an MDA tool generates. There are tools that generate infrastructure code but do not generate business- and application-logic code. Typical examples are tools that generate code for J2EE and .NET environments. In these cases, the amount of code still to be written can be large, depending on the complexity and size of the system. Conversely, there are tools that implement what is called Executable UML (xUML). In the xUML approach, business logic is expressed in the models as state machines and an accompanying action language. In these cases, the application and business logic is generated by the tool as well, minimizing the amount of code to be written. As attractive as the idea of not having to write code can be, the SEI has found that representing all business and application logic in a system as a state machine can be difficult and not intuitive for all types of systems. For example, representing a bank account as a state machine would include states such as open, closed, suspended, and overdrawn, and there are clear rules that take an account from one state to another. However, representing a bank customer as a state machine is not so clear because there are potentially only two states, active and inactive, and what takes a customer from one state to another depends on his or her bank account. Also, the action language provided by these xUML-based tools is not complete. For example, if the business logic requires complex mathematical operations, these would have to be provided by external code.

Another element that factors into development time, especially the first time that the tool is used, is the configuration of the model-to-code translators when there are mismatches between the translator and the target environment. Most MDA tools provide guidance on how to modify these translators or rewrite them completely. For example, if the tool provides a C translator for a stand-alone environment, the translator would have to be largely modified or rewritten completely if the target is C language for a distributed environment. The SEI has found that writing a translator or modifying an existing one is not an easy task. During the configuration of the translators, a lot of trial and error is involved in looking at the generated code to analyze any problems, changing configuration parameters in the tool, and repeating this process until the application works. This obviously requires understanding the generated code and the execution platform at least at a conceptual level.

So if the model-to-code translators need to be written entirely or heavily tailored, the development time will not be reduced, or might even be increased, for the first application for which MDA is used. It is likely that development time will go down for the generation of future applications running on the same platform, but this is still something that requires further study. A large initial investment in tool configuration may be reasonable in industry where there is usually a common deployment platform, but in the military where the same application is deployed on multiple platforms, this might be a problem.

The Future of MDA

MDA is a concept that has to be fully embraced and implemented by tools before it can become a widespread and cost-effective practice. Most of the tools that the SEI has

investigated are model-to-code generators for one specific platform that use UML to represent models, but the internal representation of these models is specific to the tool and therefore cannot be shared with other tools. The good news is that OMG continues to make progress in its specifications. Until tool vendors start fully conforming to these specifications, an MDA tool is nothing more than an automated code generator, and that is not where the real benefit of MDA lies.

References

[Lewis 04]
Lewis, Grace A. & Wrage, Lutz. Approaches to Constructive Interoperability (CMU/SEI-2004-TR-020). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2004.

[OMG 03]
Object Management Group. MDA Guide Version 1.0.1., 2003.

New Curriculum Fosters Adoption of Software Product Line Practices

NEWS AT SEI

Author

Paul C. Clements

This library item is related to the following area(s) of work:

Software Product Lines

This article was originally published in News at SEI on: February 1, 2005

A software product line is a set of software-intensive systems that share a common, managed set of features satisfying a particular market or mission area, and that are built from a common set of core assets in a prescribed way. Organizations that leverage the core assets (software, designs, documentation, test artifacts, budgets and schedules, tools, and more) across the family of systems they produce (instead of building each system separately) are able to dramatically increase quality and reduce cost and time to market. But adopting a product line approach to software is a technical and a business decision that involves many challenges.

The SEI has long been active in the field of software product lines, helping organizations to master the skills necessary to enjoy the high payoff possible with the product line approach. Now the SEI has introduced a comprehensive new curriculum that is designed to train software managers and practitioners in this new and powerful development paradigm. The SEI Software Product Line Curriculum is based on extensive SEI and community experience in developing and acquiring software product lines. The curriculum is supported by a widely acclaimed practitioner book in the SEI Addison-Wesley Series (*Software Product Lines: Practices and Patterns*, by Paul Clements and Linda Northrop) as well as leading-edge reports, case studies, and product line artifacts.

The curriculum includes five two-day courses:

Software Product Lines: This course provides an introduction to the world of software product lines. It covers the essential technical and management practices needed to

Related Links

Training

Software Product Lines - eLearning

Software Product Lines

See more related courses »

succeed with software product lines, and provides guidelines and patterns helpful in practical application of the product line techniques. The course covers costs and benefits of a product line approach, the technical and organizational management issues involved in the product line approach, and a number of important practice areas critical to achieving product line success. In-depth case studies, drawn from real-world experiences in government and industry, illustrate the concepts.

Adopting Software Product Lines: The benefits of taking a software product line approach are well documented, but there can be considerable barriers to organizational adoption of product line practices. This course begins with a discussion of software product line adoption issues and then presents a phased approach to treating product line adoption as a technology change. The course also presents an explicit pattern-based product line adoption roadmap that is examined in detail in terms of focus areas, phases, sub-patterns, and related practice areas, outputs, and roles. The course also describes strategies for creating synergy within an organization between product line adoption and other ongoing improvement initiatives, such as Capability Maturity Model Integration (CMMI).

Developing Software Product Lines: Developing a software product line requires planning, orchestration, and technical expertise. For those who have never been part of a working product line organization, the details of and interrelationships among the artifacts can be intimidating. This course provides hands-on experience in understanding and applying the practice areas needed for software product line mastery. It provides an in-depth treatment of the concepts and essential ideas covered in the Software Product Lines course. Using a comprehensive software product line example that includes a complete set of assets and artifacts, participants will work together to carry out many of the management and engineering activities necessary for successful product line practice.

PLTP Team Training: The SEI Product Line Technical Probe(PLTP) is a method for examining an organization's readiness to adopt or ability to succeed with a software product line approach. The PLTP is a diagnostic tool that uses the SEI's Framework for Software Product Line Practices as a reference model, comparing framework practices to the practices within the organization. The result is a set of findings that portray an organization's strengths and challenges with regard to a product line approach. This course prepares participants to be team members on a PLTP. The course will involve several hands-on exercises so that participants will learn the process and practice the skills necessary to be a contributing PLTP team member.

PLTP Leader Training: Leading a Product Line Technical Probe (PLTP) requires an in-depth understanding of product line technical and management practices, a thorough understanding of the PLTP process, excellent organizational skills, and superior facilitation skills. This course prepares participants to lead a PLTP. The course provides several hands-on exercises so that participants will learn the PLTP process from the leader's point of view and practice the social and technical skills necessary to lead a PLTP.

Software professionals can take individual courses based on specific needs or interests, or complete one or more of three specifically designed certificate programs that will be available as part of the Software Product Line curriculum.

- *Software Product Line Professional Certificate Program*: This three-course sequence provides the knowledge needed to professionally apply software product line practices.
- *Product Line Technical Probe Team Member Certificate Program*: This four-course sequence prepares a qualified software professional to perform SEI-authorized Product Line Technical Probes.
- *Product Line Technical Probe Leader Certificate Program*: This five-course sequence (plus a field observation exercise) provide a qualified software professional with the technical depth and social techniques needed to effectively lead SEI-authorized Product Line Technical Probes.

The curriculum is produced by the SEI's Product Line Systems Program, which is also

(< 5 minute) survey.

home to the SEI's successful Software Architecture Curriculum. Visit our site for information about the Software Product Line Curriculum.

Find Us Here

Share This Page

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

Using the People Capability Maturity Model with CMMI

NEWS AT SEI

Author

Suzanne D. Couturiaux

This library item is related to the following area(s) of work:

CMMI

Process Improvement

This article was originally published in News at SEI on: February 1, 2005

In the 10 years since Version 1 of the People Capability Maturity Model (People CMM) framework was published, organizations in the United States and around the world have been using it as a guide to improve their workforce practices. The People CMM, which is compatible with the staged representation of Capability Maturity Model Integration (CMMI) models, is a five-stage model that helps organizations address their critical people issues [Curtis 02]. "It looks at the knowledge, skills, and process abilities you need to run the process and run the business," says Gian Wemyss, a member of the SEI People CMM team. "It's a way to incorporate the *people* (human capital and human resources) aspects with process and technology."

Benefits Achieved

"There's a wonderful synergy when using the People CMM with CMMI," states Sally Miller, an author of the People CMM and member of the People CMM team. "CMMI describes managing your project to get the work delivered, and the People CMM describes managing your people to ensure you have the right talent at the right place at the right time." Many organizations—such as Boeing and Lockheed Martin—are tying the People CMM with CMMI as part of their overall process improvement effort. Others—such as Novo Nordisk in Denmark and Intel—are tying the People CMM with their strategic planning. "They use the People CMM to bring improvement to the whole organization," says Wemyss. These organizations have reported a number of benefits—such as a decrease in voluntary employee turnover, an increase in employee satisfaction, and an increase in productivity—from using the People CMM to manage

Please note that current and future CMMI research, training, and information has been transitioned to the CMMI Institute, a wholly-owned subsidiary of Carnegie Mellon University.

Related Links

News

SEI to Co-Sponsor 26th Annual IEEE Software Technology Conference

SEI Cosponsors Agile for Government Summit

Training

See more related courses »

Events

Team Software Process (TSP) Symposium 2014

Nov 3 - 6

and develop their workforces. For example, Boeing IT Services, World Headquarters Support (which was assessed at Level 3 of the People CMM) is integrating its People CMM activities with company-wide CMM activities and has reported a 15% increase in its employee satisfaction index, a 50% reduction in post-release defects, and a 15% reduction in rework ratio [Foster 05]. In 2002, Lockheed Martin Mission Systems (LM-MS), which was assessed at Level 3 of the People CMM in 1999, reported an annualized attrition of less than 6% and a steady reduction in its attrition rate [Miller 02]. LM-MS has achieved Level 5 of the Software CMM and is using the People CMM to help sustain higher CMMI maturity levels via solid workforce practices [Curtis 02].

Some of the benefits that organizations achieve from using the People CMM are qualitative in nature. Wemyss says, "The People CMM demands tying your workforce improvement to both your strategic plan and your operational plan, so you have a much clearer coupling of what your workforce is doing in relation to the plans and direction of the organization. It becomes clear how each person's knowledge, skills, and process abilities support the organization's operational and strategic plans."

"Organizations have also reported a greater linking between the human resources department and technical managers," states Wemyss. "Because of the way the People CMM describes the practices, organizations understand what is a manager's responsibility and what is human resources' responsibility." The People CMM enables technical managers and the human resources department to use a common language about workforce issues. Miller states, "Managers and the human resources group are able to talk about workforce issues in a more collaborative way, take measurements on how the organization's practices are working, and use the measurements to improve predictability and take inventory." For example, many organizations face an aging workforce. Organizations need to take inventory of the knowledge and skills possessed by the employees who will be retiring and transfer that intellectual capital to the people who will remain in the organization. "The People CMM helps an organization think about what it's going to do in the future when the senior employees retire," says Miller.

Applying the People CMM with a Process Improvement Effort

When applying the People CMM together with CMMI, it is important to present the People CMM as a component of the organization's overall process improvement effort and not as a separate program or a human resources program alone. Organizations should blend their workforce development and process improvement activities, integrate improvements into the organization's operational fabric, and develop a strong partnership between Human Resources and technical management. Miller adds, "Organizations that have been successful with either or both CMMI and the People CMM have made the models their own and incorporated them into the way they do their work on a regular basis. They've taken the framework provided in the models and built it into their own standards." When introducing multiple improvement programs, an organization needs to assess the amount of change it can absorb and stage the introduction of the programs. For example, if an organization is at CMMI Level 2, project managers should first master their project management skills and then begin improvements guided by the People CMM to supplement their project management activities [Curtis 03].

Plans to Update the People CMM Product Suite

The People CMM was originally developed by the SEI with support from the U.S. Army and Office of the Secretary of Defense. Until June of 2005, TeraQuest Metrics (a process consulting firm) had responsibility for the commercialization of the People CMM and support of authorized People CMM assessors and instructors. With the recent acquisition of TeraQuest Metrics by Borland Software Corporation, these activities will now be handled by the SEI. Wemyss states, "This will result in a greater tying of the People CMM with CMMI. The People CMM product suite is going to parallel the CMMI product suite." The courses offered by the SEI for the People CMM will be analogous to the courses offered for CMMI. Currently, there is an introduction course and an intermediate concepts course for CMMI, but there is only an introduction course for the People CMM. In 2006, the SEI plans to offer a new Intermediate Concepts of People CMM course. In addition, the SEI plans to update the appraisal method from the People CMM-based assessment method to the Standard

CMMI Appraisal Method for Process Improvement (SCAMPI) method and will announce SCAMPI training and instructor training for the People CMM.

The new intermediate course is implementation focused and will give attendees a better idea of how to apply the model. Wemyss states, "The current Introduction course is about getting to know what's in the model; with the new intermediate course you can focus on a deeper knowledge of the model and on how to implement what's in the model in your own organization." With the addition of this new course, the training for people who want to become People CMM appraisers or instructors will parallel the training for CMMI appraisers and instructors. Miller explains, "The path you'll follow for the People CMM will be similar to the path you'll follow for CMMI. The difference is the People CMM covers workforce practices, and CMMI covers project practices." The first pilot offering of the Intermediate Concepts course is scheduled for September 12-16, 2005 at the SEI.

Currently, the People CMM based assessment method is used for appraisals against the People CMM, and SCAMPI is used for appraisals against the CMMI framework. The SCAMPI appraisal method will allow appraisal teams to use one method for both People CMM and CMMI appraisals. With this change, appraisal teams will no longer need to learn two appraisal methods and organizations will be able to use lower cost appraisals (Class B and C) that don't result in a maturity rating to determine if their workforce improvement efforts are succeeding. Wemyss says, "There was a pilot earlier this year at Tata Consultancy Services—India's largest system and software services organization—where SCAMPI was used against the two models. It saved money because you didn't need two teams, you didn't need to train the team in two different methods, and you could do two appraisals at once."

References

[Curtis 02]
Curtis, Bill; Hefley, William E.; and Miller, Sally A. *The People Capability Maturity Model: Guidelines for Improving the Workforce*. Boston, MA: Addison-Wesley, 2002.

[Curtis 03]
Curtis, Bill; Hefley, William E.; and Miller, Sally A. "Experiences Applying the People Capability Maturity Model." *Crosstalk* (April 2003).

[Foster 05]
Foster, Ken. "Utilizing the People CMM to Leverage Organization Success." *Proceedings of Software Engineering Process Group (SEPG) Conference 2005* (CD-ROM). Seattle, WA, Mar. 7-10, 2005. Pittsburgh, PA: Carnegie Mellon University, 2005.

[Miller 02]
Miller, Cecilia. "Practical Approaches to Initiating and Sustaining a Successful People CMM Effort." *Proceedings of Software Engineering Process Group (SEPG) Conference 2002* (CD-ROM). Phoenix, AZ, Feb. 18-21, 2002. Pittsburgh, PA: Carnegie Mellon University, 2002.

Using CMMI with Suppliers – The Acquirer's Concerns

NEWS AT SEI

Author

Mike Phillips

This library item is related to the following area(s) of work:

CMMI

Process Improvement

This article was originally published in News at SEI on: February 1, 2005

Two previous columns have focused on appraisals and it is time for another installment. I'll begin with a brief history of the general use of CMM-based approaches applied not for internal process improvement, but by external entities. These are usually government organizations seeking to choose or otherwise examine a potential (or existing) provider of software-intensive products.

A Short History

Many of you may know that the CMM actually began its life as a way for government program managers to gather questions so they could gain confidence in selecting contractors to provide software-intensive systems for the Air Force's Electronic Systems Center. Over time these questions became a set of best practices to look for and then became the SW-CMM model and now CMMI. From an appraisal perspective, these began as Software Process Assessments (SPAs). The SPA was replaced by two approaches, the CMM-Based Appraisal for Internal Process Improvement (CBA IPI) and the externally focused Software Capability Evaluation (SCE). With CMMI, the SEI was asked by the DoD to assure that one method, SCAMPI V1.1, would be applicable to either internal or external use.

The establishment of a level 3 expectation as a discriminator has its own history. The first "requirement" for level 3 was established by a key Air Force executive in the Pentagon. This memorandum, though, was directed to government organizations that wished to provide organic capability, and insisted that level 3 was the expectation.

Please note that current and future CMMI research, training, and information has been transitioned to the CMMI Institute, a wholly-owned subsidiary of Carnegie Mellon University.

Related Links

News

SEI to Co-Sponsor 26th Annual IEEE Software Technology Conference

SEI Cosponsors Agile for Government Summit

Training

See more related courses »

Events

Team Software Process (TSP) Symposium 2014

Nov 3 - 6

Over time level 3 grew to become a common request in many requests for proposal, both inside and outside of government. For a little over a year, DoD had a policy requiring CMM level 3 for its largest (ACAT I) programs, but a revision to DoD directives eliminated the policy guidance. While specific guidance about these expectations is gone, encouragement of appraisals remains. There are two paragraphs covering this in DoD 5000.1:

4.2.5.2 Capability Reviews. Capability reviews such as manufacturing capability and software capability reviews are a useful tool available during source selections to assess the offerors' capability in selected critical process areas. Capability reviews may be the appropriate means for evaluating program-specific critical processes such as systems engineering, software development, configuration management, etc. The reviews would be useful to supplement process past performance data to ascertain the risks in selecting a given offeror and to assist in establishing the level of government oversight needed to manage the process-associated risks if that offeror is awarded the contract. The trade-off in determining whether or not to do a capability review would be the criticality of the process versus the time and resources to do the review versus the availability, adequacy, and currency of an offeror's process past performance data.

4.2.5.3 Capability Appraisals. In all cases, the program manager retains the right (and is encouraged) to independently evaluate the process capabilities of the selected team prior to or immediately after contract award in order to have a better understanding of potential risks associated with the development team's process capabilities. Once the developer is selected, the program manager can conduct an evaluation to support the up-front risk assessment of the developer's capability to deliver. Periodic appraisals are encouraged as part of contract process monitoring activities. The selection of assessment or appraisal method would be dependent upon the needs of the particular project, the level of risk associated with the project, and any areas of concern the program manager may have. The program manager should understand that: 1) appraisal and assessment results are another tool (like past performance) to gauge the likelihood that the contractor will succeed and perform to the requirements of the contract; 2) assessments are most valuable when they apply across the full program team, and not just one segment of the organization; and 3) domain experience is at least as important as process maturity level when evaluating the program team's capability.

Maturity Levels and Program Success

Maturity levels are an imperfect approach to assuring program performance. Maturity levels are indicators of organizational potential: they describe how the next project will most likely be conducted based on a sampling of existing projects. Maturity levels describe the organization and are not an indication of how an individual project is performing. The organizational scope of an appraisal is usually limited to a division or company within a larger corporation that manages similar projects or a product line (see Figure 1). This "organization" may have many projects managed within its scope, but for the purposes of an appraisal, the number of focus projects is usually limited to between three and six. These focus projects must, taken as a whole, represent the entire lifecycle and be representative of the implemented processes and functional areas being investigated within the organizational unit.

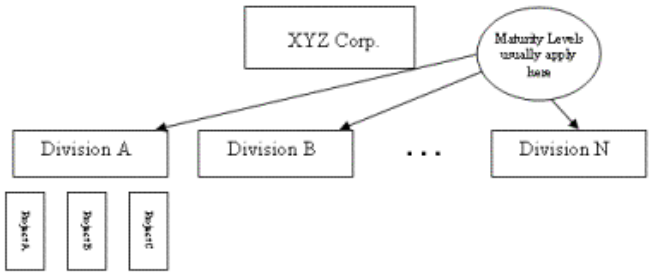

Figure 1: Applicability of CMMI Maturity Levels

Acquisition programs are concerned with the performance of their programs. The program is usually made up of a set of projects from multiple contractor organizations,

teamed with the acquisition office to deliver capability to an end user (see Figure 2). Choosing contractors with high maturity levels is necessary but not sufficient to ensure that strong systems-engineering and program-management practices are employed on any program.

Figure 2: The Acquirer's Concern

A better way of ensuring strong systems-engineering and project-management practices is to evaluate the capability of potential bidders in select areas of importance to an acquisition program. For example, if an acquisition program is looking to hire a lead systems integrator, the areas of concern may be the bidder's ability to manage risk, manage suppliers, plan and track the program, build an integrated team, develop an architecture, and integrate the various components. During the source selection, the acquirer could then evaluate the capability of these specific areas using relevant CMMI process areas to determine the strengths and weaknesses of a given bidder. This approach provides a key discriminator in the selection process against a publicly available model and a set of industry accepted practices.

Recent Developments

In recent years, a government-industry team concluded that it would be helpful to establish a registry of appraisals done by industry with government participation on the appraisal teams. Such a registry was viewed as providing government more confidence in the outcome. This approach remains available, with the SEI maintaining the record of government membership. If queried, the SEI can provide a reference to the participants without compromising the confidentiality of the actual appraisal data.

Although the addition of external evaluation options was provided with the release of SCAMPI V1.1 and an associated guide for use in source selections and contract monitoring, the SEI has not received any of these appraisals to date. Existing government evaluators have noted that a "full" (level 3) appraisal, performed on the set of proposing contractors, would take too long to be acceptable as part of a source selection.

Over the last two years, the SEI teamed with a consortium of organizations using CMMI and a set of government evaluators to collect some of the best practices for the less-robust appraisals that we have characterized as class "B" or "C." We had determined that our SCAMPI Lead Appraisers deserved to have a tool-kit of building-block appraisal approaches that were compatible with existing "SCAMPI A" appraisals.

The handbook that has resulted from this work includes guidance that a government team would need to have to conduct an externally oriented appraisal that uncovers needed source selection (or contract monitoring) information. The handbook helps teams gather strengths and weaknesses of the program-development team and likely risks to program execution, without seeking achievement of any specific maturity level. And because we have maintained the requirements for CMMI knowledge at the same level that we expect of our SCAMPI A teams, we believe we can maintain the needed confidence in appraisal-team performance. This clearly is a concern for both the government and contractor sides of these external appraisals.

Many acquisition programs are currently using this handbook, including the Navy's Multi-Mission Maritime Aircraft program, the National Reconnaissance Office's Future Imagery Architecture Program, the National Security Agency's Cryptologic Mission Management program; it has also been adopted by Australia's Defense Materiel

Organization.

After contract award, the acquisition program can then continue to evaluate the actual performance of the entire team, including the acquisition program office, against select process areas of interest from the CMMI for the developers and the CMMI Acquisition Module (CMMI-AM) for the acquisition program. This ongoing look at how the program is performing can provide early indicators of process-related risk and will help ensure that strong systems-engineering and project-management practices are employed across the entire program team.

About the Author

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

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

Governing for Enterprise Security: Security is a Requirement of Being in Business

NEWS AT SEI

Author

Julia H. Allen

This article was originally published in News at SEI on: February 1, 2005

Enterprise security is crucial to almost all organizations. But with so many other topics vying for your attention, what priority should you assign to enterprise security? What constitutes adequate security and thus adequate governance? How much security governance is enough and how can you use governance to sustain adequate security in a constantly changing business, risk, and technology environment?

A recent Business Roundtable report, *Securing Cyberspace: Business Roundtable's Framework for the Future* [BRT 04] asserted that

- Information security requires the CEO attention within individual companies as business leaders seeking collectively to promote the development of standards for secure technology.
- Boards of directors should consider information security an essential element of corporate governance and a top priority for board review.

Governance involves careful oversight and well-informed decision making. The resulting actions set expectations for an organization's conduct. *Governing for enterprise security (GES) means that security is viewed as a requirement of being in business*. GES must be addressed at the leadership level and not be relegated to a technical specialty within the IT department. The role of boards of directors, executives, and senior managers must be to establish and reinforce the business need for effective enterprise security. Otherwise, the organization's desired state of security will not be articulated and thus cannot be achieved or sustained. If the responsibility for enterprise security is relegated to a role in the organization that lacks the authority, accountability, and resources to act and enforce, the enterprise security state will reflect this and remain far below an optimum level.

Leaders need to understand that business objectives must guide and drive actions

needed to govern for enterprise security. The connection is clear when you examine a list of organizational "assets" that can be negatively affected if GES is performed poorly:

- trust
- reputation
- brand
- shareholder/stakeholder value
- marketplace and stock market confidence
- customer retention and growth
- customer and partner identity and privacy
- ability to offer and fulfill electronic transactions

Leaders need to evaluate how much their enterprises depend on Internet connectivity, information technology (IT) infrastructure, and electronic assets for business continuity. Then, they can better determine the degree to which governance decisions need to account for the security of such assets. Factors that can aid in making this determination are described below.[1] Together with a good risk assessment, an aggregation of these factors can inform your security-investment decisions.[2]

Organization Characteristics

- size (number of physical locations, employees, and customers; level of revenue)
- complexity (organizational units, products, services, processes, systems, partnerships, structure—e.g., centralized or decentralized)
- value/criticality of the organization's intellectual property stored or transmitted in electronic form
- dependence on IT systems and the Internet to offer products and services to customers
- impact of major system downtime or an Internet outage on the organization
- degree of change within the organization (expansions, mergers, acquisitions, divestitures, new markets, etc.
- dependence on multinational operations
- plans for multinational operations (internal functions outsourced to offshore locations, moving into geographical areas representing increased portions of overall revenue)

Market Sector Characteristics

- potential impact to national, international, or critical infrastructures as a result of outages or interruptions in organizational systems
- customer sensitivity to security and privacy
- level of sector regulation that addresses security [e.g., Gramm-Leach Bliley Act (GLBA), Health Insurance Portability and Accountability Act (HIPAA), Sarbanes Oxley[3], other applicable international, national, state, or local regulations]
- potential brand and reputation impact of a publicly disclosed security incident
- extent of enterprise operations dependent on third parties (partners, contractors, suppliers, vendors) and connectivity with third-party networks
- customers' ability to quickly switch to a competitor, based on competitor's ability to offer more secure, reliable services
- extent to which the organization does business in a politically sensitive area where it could be a likely target of violent physical or cyber attack

Organizations will be far ahead if their leaders treat the governance of enterprise security as essential to their businesses and are aware and knowledgeable about the issues. Ultimately, nations as a whole benefit: "The critical information infrastructures

comprising cyberspace provide the backbone for many activities essential to the transaction of domestic and international business, the operation of government, and the security of a nation." [BRT 04]

Dan Geer, in his *Cutter Consortium Business-IT Strategies* article titled "Why Information Security Matters" [Geer 04] states:

"The central truth is that information security is a means, not an end. Information security serves the end of trust. Trust is efficient, both in business and in life; and misplaced trust is ruinous, both in business and in life. Trust makes it possible to proceed where proof is lacking. As an end, trust is worth the price. Without trust, information is largely useless."

The next article in this series will discuss the shifts in perspective that are required for leaders to achieve and sustain enterprise-wide security.

References

[BRT 04]
Business Roundtable. "Securing Cyberspace: Business Roundtable's Framework for the Future." May 2004.

[CGTF 04]
Corporate Governance Task Force. "Information Security Governance: A Call to Action." National Cyber Security Partnership, April 2004.

[Geer 04]
Geer, Daniel E. "Why Information Security Matters." *Cutter Consortium Business-IT Strategies Vol. 7*, No. 3, 2004.

[TechNet 03]
TechNet. "Corporate Information Security Evaluation for CEOs—Preview Draft." December, 2003.

1 These factors are derived from [CTGF 04], based on original work reflected in [TechNet 03]. Refer to the TechNet evaluation to see one application of these factors.

2 The terms "enterprise," "organization," and "business" are used interchangeably. "Agency" or "institution" can be easily substituted.

3 Also known as the Public Company Accounting and Investor Protection Act of 2002.

About the Author

Julia Allen is a senior member of the technical staff within the Networked Systems Survivability Program at the Software Engineering Institute (SEI), a unit of Carnegie Mellon University in Pittsburgh, PA. The CERT Coordination Center is also a part of this program.

Allen is engaged in developing and transitioning enterprise security frameworks and executive outreach programs in enterprise security and governance. Prior to this technical assignment, Allen served as acting Director of the SEI for an interim period of 6 months as well as Deputy Director/Chief Operating Officer for 3 years. Her degrees include a B. Sci. in Computer Science (University of Michigan) and an MS in Electrical Engineering (University of Southern California). She is the author of *The CERT Guide to System and Network Security Practices* (Addison-Wesley, June 2001).

The views expressed in this article are the author's only and do not represent directly or imply any official position or view of the Software Engineering Institute or Carnegie Mellon University. This article is intended to stimulate further discussion about this topic.

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

Only Leaders Need Apply

NEWS AT SEI

Author

Linda M. Northrop

This library item is related to the following area(s) of work:

Software Product Lines

This article was originally published in News at SEI on: February 1, 2005

In 1981 Jack Welch assumed the position of CEO at General Electric (GE). He quickly unveiled an aggressive strategic redirection that transformed GE into 14 distinct businesses focused on three "strategic circles": core manufacturing, technology-intensive businesses, and services. [Tichy 89] He coupled his business overhaul with an initiative to revitalize the work force. He was passionate about his vision, and he relentlessly drove it to completion. But he didn't stay at the vision and strategy level of organizational management; he got involved in GE daily operations. In fact he never hired a chief operating officer (COO) despite the immensity of the enterprise he ran. He interacted directly with GE workers at all levels. He believed in his staff, listened to them, and gave them abundant feedback. He made them part of his solution, and embraced their good ideas. He broke down the walls between managers and subordinates. He was interested not only in his success but in GE's success. He spent years selecting, nurturing, and pruning a set of candidate successors. Welch's innovative strategies and management style transformed GE into a "highly productive, labor-efficient powerhouse with a staggering $200 billion-plus market capitalization." [Slater 98] Under Welch GE soared to the top of the Fortune 500.

My brother has been a high school football coach for over 35 years. His teams have amassed 19 championship and five unbeaten seasons. The high school even named the football stadium after him. He has encyclopedic knowledge of football and athletic ability to match, but it's more than his knowledge and skills that have made him a legend. He does all of the workouts with his team and keeps the training rules he sets for them. He gets to know each player and relentlessly drives them all to achieve their potential and then some. He is always in the game, not just directing it. One season he "fired" more than half of his first-string players for breaking in-season training rules.

Related Links

Training

Software Product Lines - eLearning

Software Product Lines

See more related courses »

(< 5 minute) survey.

Despite vehement pressure from parents, the school board, and administrators, he refused to reverse his decision. As a result, the team was purged of its finest talent, and some of the remaining players had to play both offense and defense. He worked tirelessly with the skeleton squad, challenging them and spurring them on. He believed they could succeed, but more importantly, he made them believe they could succeed. And they did; they went to the championship playoffs.

Now what could Jack Welch and my brother possibly have in common? Simple: they are both leaders. And what do leaders have to do with software product lines? Everything: you can't just manage a software product line operation, you have to lead it. Managers control. Leaders spur an emotional energy that fuels an organization to succeed despite downturns and obstacles. Ron Temple led Cummins Engine Co. boldly through turbulence to success with its product line approach for engine-control software. Jaak Urmi did the same for CelsiusTech. In fact, there is a leader behind each story of successful product line adoption that we have studied at the SEI. And of the failed product line efforts we have seen, most either lacked a leader or lost one.

If product line management must lead to succeed, then we ought to take time out to describe the qualities of a leader. We have only to study the role models such as Welch, my brother, Temple, and others. Leaders are

- intelligent. They know their business. They can quickly assess a problem and arrive at a solution.
- self-assured. They are confident in their abilities and don't need continual gratification. They surround themselves with top-notch talent and are not threatened by good ideas and brilliance.
- visionary. They can see the whole picture and can project it into the future. They see trends and can imagine solutions beyond what exists or what others think can exist.
- enthusiastic. They have an unquenchable, contagious zest for their work and their organization.
- competitive. They are hungry; they push and push to make things happen.
- decisive. They are not afraid to make decisions and do so quickly.
- committed. They believe in their goals and focus to achieve them.
- flexible. They can change gracefully if a change is needed.
- articulate. They are good communicators.
- people focused. They have a deep interest in people and care about the people in their organizations.
- realistic. They can recognize when things are not working or when a change in approach is needed. They strive for the maximum effort but not the impossible.
- externally focused. They are in tune with the world around them.
- open. They don't need to hide behind a cloak of secrecy.
- honest. They tell the truth and expect the same of others.
- trusting. They empower and have faith in others.
- action-focused. They get things done. Leaders have what CEO coach Ram Charan calls "emotional strength" [Tichy 89].

Given these qualities, what do leaders in product line settings do that sets them apart? They

- Pave the way.
- Understand what it takes to succeed with product lines.
- Know their business and their competition.
- Establish a product line vision and tangible product line goals; communicate both clearly and often.
- Take risks but have mitigation strategies in place.

- Garner support (including stable funding) from their superiors, from customers, and from boards of directors and keep their superiors informed and enthused about product lines.
- Know their people, assign them to jobs that match their talents, and hold them accountable.
- Train their people so that each is knowledgeable in his or her product line role.
- Act as the product line champions or recognize and support those who do.
- Take an active role in the journey.
- Execute their plans and deliver on commitments.
- Make decisions quickly and communicate them openly.
- Use process to drive decisions and accomplish work, not to delay it.
- Drive the organization to succeed but reward achievements along the way.
- Link pay directly to performance that supports the product line goals.
- Keep track of all critical assignments.
- Know what's going on in their organizations. They pay spontaneous visits to workers at all levels. They listen to the undercurrent and pay attention to morale.
- Confront reality. If a strategy isn't working, they change it. If people aren't performing, they coach, reassign, or fire.
- Believe in their people. They listen and provide feedback.
- Don't get discouraged by the inevitable backlash to a technology adoption.
- Know that they are there to support their people, not the other way around.

This is a long list that goes beyond what most would call management. Jack Welch refused to call himself a manager. He preferred the term "business leader." To him a leader is a teacher, a cheerleader, and a liberator, not a controller [Tichy 89]. If you are choosing the individual to be at the helm of a product line effort, choose a leader. If you are the person in charge, take heed.

References

[Slater 98]
Slater, R. *Jack Welch and The G. E. Way: Management Insights and Leadership Secrets of the Legendary CEO*. New York, NY: McGraw-Hill, 1998.

[Tichy 89]
Tichy, N. and Charan, R. “Speed, Simplicity, Self-Confidence: An Interview with Jack Welch.” *Harvard Business Review*, September-October 1989, Number 5: 112-121.

About the Author

Linda Northrop has over 30 years of experience in the software development field as practitioner, manager, consultant, and educator. She is currently director of the Product Line Systems Program at the Software Engineering Institute (SEI). The Product Line Systems Program works in the areas of software architecture, component, and product line engineering. Her current publications are in the areas of software product lines, software architecture, and object technology. She is co-author of the book, *Software Product Lines: Practices and Patterns.*

She is a frequently invited speaker at technical and corporate conferences and was featured in a television special on object technology aired by the British Broadcasting Company. Most recently she was a keynote speaker for the International Conference on Software Engineering 2001, the Aspect-Oriented System Development Conference 2002, and was selected as the 2001 Carnegie Science Center Award of Excellence for Information Technology recipient.

Before joining the SEI, she was associated with both the United States Air Force Academy and the State University of New York as professor of computer science, and with both Eastman Kodak and IBM as a software engineer. As a private consultant, she also worked for an assortment of companies covering a wide range of software systems. She chaired the first Software Product Line Conference (SPLC1) in 2000 and

SPLC2 in 2002. Linda is the OOPSLA Steering Committee Chair and was OOPSLA 2001 Conference Chair. She is a member of ACM and the IEEE Computer Society, and formerly the Computer Sciences Accreditation Commission.

The views expressed in this article are the author's only and do not represent directly or imply any official position or view of the Software Engineering Institute or Carnegie Mellon University. This article is intended to stimulate further discussion about this topic.

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

Architecture Business Cycle Revisited: A Business Goals Taxonomy to Support Architecture Design and Analysis, The

NEWS AT SEI

Authors

Len Bass

Paul C. Clements

Rick Kazman

Robert Nord

This library item is related to the following area(s) of work:

Software Architecture

This article was originally published in News at SEI on: February 1, 2005

Related Links

News

SATURN Conference Announces Additional Keynote, Conference Scholarships

Distinguished Speakers, Strong Technical Program Set for SATURN 2014

See more related news »

Training

Big Data - Architectures and Technologies

Documenting Software Architectures - eLearning

See more related courses »

Business Goals as *Forces*

The elicitation of a software-intensive system's business goals is an integral part of standard architecture design and analysis methods. Business goals are the engines that drive the methods, through their realization as scenarios. That is, business goals are forces that shape the architecture. The architecture can be designed to make a system that is, for example, high performance, or easy to modify, or easy to integrate with other systems, or highly reliable, where each of these is presumably supporting a business goal (for example, a system supporting an e-commerce Web site should be high performance, highly reliable, secure and so forth). This architectural "shape" be achieved only when the business goals are translated into scenarios, which can be mapped onto architectural structures that realize the goals.

In the past, despite its crucial importance as the driver of architectural quality, the elicitation of business goals has typically been an unstructured activity. In an attempt to structure this activity, we have created a taxonomy of business goals. The idea of the taxonomy is to facilitate the thinking and to stimulate the creativity of stakeholders,

and to ensure proper coverage of all business issues and all the forces that bear on an architecture.

An ideal design method or process must take into account *many* stakeholders and *many* forces. These forces transcend the traditional realms of requirements specification, and include business and political issues, legal or contractual issues, life-cycle concerns, the competitive environment of the system, the experience of the architect, and the goals of the developing organization. To our knowledge, no design method attempts to address all of these concerns. There is likely a good reason for this: such a method would probably be unwieldy. However, system stakeholders still face all of these concerns, and so, when choosing a design method, they should be able to choose one that focuses on the concerns that concern them the most. Currently we are attempting to enumerate these forces and to place them into a system-development context.

We have organized the set of forces into the following seven categories as follows:

1. stakeholder needs
2. business management issues
3. legal/contractual issues
4. commercial/competitive pressures
5. technical environment
6. political issues
7. life-cycle issues

The ABC Revisited

To represent the ways in which a force may affect a design, we have adopted and adapted the Architecture Business Cycle (ABC) [Bass 03]. The ABC was originally envisioned as a means of depicting the influences on an architect and on how an architecture can eventually influence the forces that influenced the architecture, thus creating a cycle. The original influences were the stakeholders and the developing organization (who, together, fashioned the quality-attribute requirements), the technical environment, and the architect's experience.

In our adaptation of the ABC, the "influences" have been replaced by a set of forces that collectively shape the requirements and, over time, the architect's experience. The extended ABC is depicted in Figure 1.

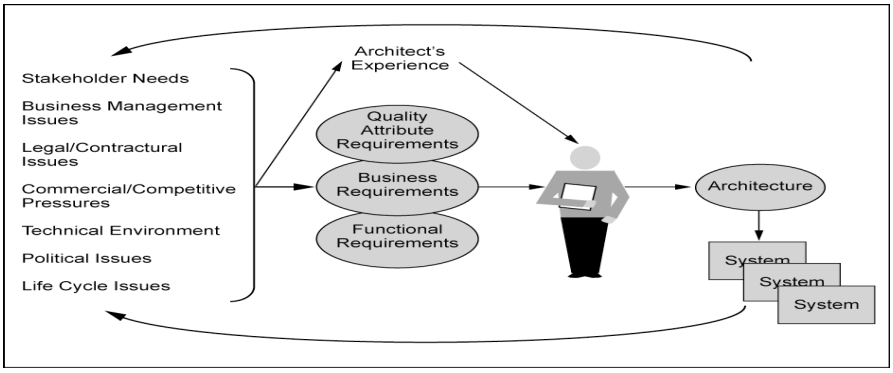

Figure 1: Extending the Architecture Business Cycle to Include All Design Forces

Notice that this extended ABC shows the requirements as being categorized into three overlapping sets: quality-attribute requirements (e.g., the system should initialize itself in 60 seconds), business requirements (e.g., version 1.0 must be available to our resellers by August 1), and functional requirements (e.g., the system must provide monthly sales summaries, broken down by region). These requirements, along with the architect's experience, are the inputs to the architect and to the architecture-design process.

Capturing a Force

If we are to propose forces that designers and design methods can consider when designing a system, we should be able to describe these forces clearly. Our goal is to answer this question:

What do we need to communicate about a force so that everyone will understand it?

We propose the following scheme:

1. stimulus source: who brings the force to bear?
2. stimulus: what is the force?
3. artifact stimulated: what does the force act on?
4. environment: when is the force brought to bear (e.g., during development)?
5. response: what does the stimulus source desire as an outcome?
6. response measure: how will the stimulus source know when the outcome has been satisfactorily achieved?
7. importance: how much does the stimulus source care about the outcome? What is its achievement worth, or what would failing to achieve it cost?

For example, we have said that efficient use of staff is a force. How would we capture that?

1. stimulus source: project manager responsible for development and operations manager responsible for system operation, who want to optimize staffing levels
2. stimulus: the force is desire to use staff efficiently and minimize new hiring. Staff include designers, developers, maintainers, deployers, operators.
3. artifact stimulated: development organization
4. environment: the force is manifested during inception, elaboration, construction, transition, production, retirement
5. response: the desired outcome is no new hiring
6. response measure: number of people, level of skill and understanding (Cockburn expert levels), total cost of employment
7. importance: cost of hiring new people, possible cost of laying off current staff lacking needed skills

This taxonomy of business goals and the template for capturing them are a first step in enriching the architecture design methods being created at the SEI and elsewhere, to make them more capable of capturing *all* the forces on an architecture and not just the technical ones that have long been our focus.

References

[Bass 03]
Bass, L.; Clements, P.; & Kazman, R. *Software Architecture in Practice, 2nd ed*. Boston, MA: Addison-Wesley, 2003.

About the Authors

Rick Kazman is a senior member of the technical staff at the SEI, where he is a technical lead in the Architecture Tradeoff Analysis Initiative. He is also an adjunct professor at the Universities of Waterloo and Toronto. His primary research interests within software engineering are software architecture, design tools, and software visualization. He is the author of more than 50 papers and co-author of several books, including a book recently published by Addison-Wesley titled *Software Architecture in Practice*. Kazman received a BA and MMath from the University of Waterloo, an MA from York University, and a PhD from Carnegie Mellon University.

Len Bass is a Senior Member of the Technical Staff at the Software Engineering Institute (SEI) and participates in the High Dependability Computing Program. He has written two award-winning books in software architecture as well as several other books and numerous papers in a wide variety of areas of computer science and

software engineering. He is currently working on techniques for the methodical design of software architectures and to understand how to support usability through software architecture. He has been involved in the development of numerous different production or research software systems ranging from operating systems to database management systems to automotive systems.

Paul Clements is a senior member of the technical staff at Carnegie Mellon University's Software Engineering Institute, where he has worked for 8 years leading or co-leading projects in software product line engineering and software architecture documentation and analysis. Clements is the co-author of three practitioner-oriented books about software architecture: *Software Architecture in Practice* (1998, second edition due in late 2002), *Evaluating Software Architectures: Methods and Case Studies* (2001), and *Documenting Software Architectures: View and Beyond* (2002). He also co-wrote Software Product Lines: Practices and Patterns (2001), and was co-author and editor of Constructing Superior Software (1999). In addition, Clements has also authored dozens of papers in software engineering reflecting his long-standing interest in the design and specification of challenging software systems. He received a B.S. in mathematical sciences in 1977 and an M.S. in computer science in 1980, both from the University of North Carolina at Chapel Hill. He received a Ph.D. in computer sciences from the University of Texas at Austin in 1994.

Robert Nord is a senior member of the technical staff in the Product Line Systems Program at the Software Engineering Institute (SEI) where he works to develop and communicate effective methods and practices for software architecture. Prior to joining the SEI, he was a member of the software architecture program at Siemens, where he balanced research in software architecture with work in designing and evaluating large-scale systems. He earned a Ph.D. in Computer Science from Carnegie Mellon University. Dr. Nord lectures on architecture-centric approaches. He is co-author of Applied *Software Architecture and Documenting Software Architectures: Views and Beyond*.

The views expressed in this article are the author's only and do not represent directly or imply any official position or view of the Software Engineering Institute or Carnegie Mellon University. This article is intended to stimulate further discussion about this topic.

Service-Oriented Architectures as an Interoperability Mechanism

NEWS AT SEI

Authors

Grace Lewis

Edwin J. Morris

Dennis B. Smith

Lutz Wrage

This library item is related to the following area(s) of work:

System of Systems

This article was originally published in News at SEI on: February 1, 2005

Government and commercial organizations have been placing increasing reliance on interoperability among large numbers of systems for achieving important mission goals. One mechanism for both achieving interoperability and for leveraging the value of legacy systems is for organizations to expose all or parts of their legacy systems as services. A service is a coarse-grained, discoverable, and self-contained software entity that interacts with applications and other services through a loosely coupled, often asynchronous, message-based communication model [Brown 02]. A collection of services with well-defined interfaces and a shared communications model is called a service-oriented architecture (SOA). A system or application is designed and implemented as a set of interactions among these services.

SOAs allow legacy systems to expose their functionality, sometimes without making significant changes to the legacy systems. SOAs also have characteristics that make effective interoperability easier, such as loose coupling, published interfaces, and a standard communication model. However, constructing services from existing systems to obtain the benefits of an SOA is neither easy nor automatic. In fact, such a migration can represent a complex engineering task, particularly when the services are expected to execute within a tightly constrained environment.

Related Links

Training

Migrating Legacy Systems to SOA Environments - eLearning

Service-Oriented Architecture: Best Practices for Successful Adoption

See more related courses »

(< 5 minute) survey.

This article is the first of two parts. The first article outlines what service-oriented architectures are as well as general issues that need to be addressed in creating services from legacy components. It also provides a notional DoD-relevant example. The second article will discuss how to evaluate services for migration to an SOA.

Service-Oriented Architectures

We have already said that a service interacts with other services through a message-based communications model. Common communications models include

- Web services using Simple Object Access Protocol (SOAP) and Web Services Description Language (WSDL)
- Message-oriented middleware (MOM) such as IBM WebSphere MQ
- Publish-subscribe system such as Java Messaging Service (JMS)

SOAs differ from other messaging models such as DCOM (Distributed Component Object Model) or Object Request Brokers (ORBs) because services of SOAs are discoverable and coarse-grained. Services must be able to be discovered at both design time and runtime, not only by unique identity, but also by interface identity and type of service. Services are also coarse-grained; that is, they implement more functionality and operate on larger data sets, as compared to components in component-based design [Lewis 04]. A typical example of a service includes a credit card validation service.

A service consumer can

- invoke a service directly from a service provider
- use a directory service to find a service provider based on some criteria. The directory service returns the location of the service so the service consumer can invoke the service provider.
- use a service broker to pass on its request to one or more directory services.

SOA and Interoperability

In a service-oriented architecture, “interoperability” refers to the ability of the service to be invoked by any potential client of the service [Stevens 03]. There are several attributes of a SOA that make this possible:

- Common payload (the actual data that is being exchanged) and protocol: each service provides an interface that is invoked through a payload format and protocol that are understood by all the potential clients of a service.
- Published and discoverable interfaces: each service has a published and discoverable interface that allows systems to search for services that are best suited for their purposes.
- Loose coupling: services are connected to other services and clients using standard, dependency-reducing, decoupled message-based methods such as XML document exchanges.
- Multiple communication interfaces: services can implement separately defined communication interfaces. For example, a service could have a Web services adapter, an IIOP (Internet Inter-ORB Protocol) adapter, and an MQSeries adapter to serve clients of these different types.
- Composability: Because services are coarse-grained reusable components that expose their functionality through a well-defined interface, systems can be built as a composition of services and evolve through the addition of new services.

From a syntactic point of view, a service-oriented architecture is promising. The challenge lies in determining the number of adapters to implement and determining the right granularity of service interfaces, because it is not always known how systems will use the services. It is important to keep in mind that services are executed as an exchange of a service request and a service response. If service interfaces are too coarse-grained, clients will receive more data than they need in their response message. If service interfaces are too fine-grained, clients will have to make multiple

trips to the service to get all the data they need.

From a semantic point of view, service-oriented architectures by themselves do not offer any guarantees. Semantic interoperability depends on how the interfaces to a service are described and how the meaning of the information is shared with potential clients of the service. Several issues that need to be addressed include

- how to know exactly what a service offers
- the quality of service it offers

Creation of Services From Legacy Components

The most common (but not only) form of service-oriented architecture is that of Web services, in which all of the following apply: (1) service interfaces are described using Web Services Description Language (WSDL), (2) payload is transmitted using Simple Object Access Protocol (SOAP) over Hypertext Transfer Protocol (HTTP), (3) Universal Description, Discovery and Integration (UDDI) is used as the directory service [Lewis 04].

Enabling a legacy system to interact within a service-oriented architecture, such as a Web services architecture, is sometimes relatively straightforward; this is a primary attraction to the approach for many businesses. Web service interfaces are set up to receive SOAP messages, parse their content, invoke legacy code directly or through a custom component that invokes the legacy code, and optionally wrap the results as a SOAP message to be returned to the sender. Many modern development environments provide tools to help in this process, and commercial organizations are rapidly employing these environments to expose their business processes to the world.

However, characteristics of legacy systems, such as age, language, and architecture, as well as of the target SOA, can complicate the task. This is the case when migrating to highly demanding and proprietary SOAs such as those being proposed for many DoD systems.

DoD SOA Context

DoD systems have recently focused on the concept of network-centric operations: providing forces with access to integrated information from a variety of previously unconnected sources [Alberts 00]. This focus requires strong interoperability to ensure that systems work together effectively. To facilitate such interoperability, the DoD has initiated a number of projects that focus on different aspects of the infrastructure for network-centric operations. Several of these projects are developing SOAs so that command and communications (C2) applications can be built as a set of interactions between infrastructure services (e.g., communication, discovery) and services that are specific to the C2 domain (application domain services). Current and future DoD program offices have been targeted to contribute application domain services.

A notional example of such an SOA is illustrated in Figure 1.

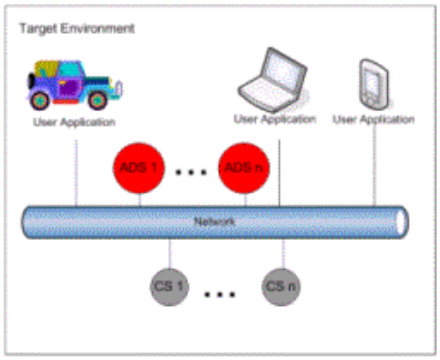

Figure 1: Physical View of Example Target SOA

Figure 1 shows that this example of an SOA includes common services (CS) that are to be used by user applications and application domain services (ADS). The SOA owns the interfaces for the common services. The environment then allows for a set of ADSs

that will derive their requirements from user applications. Groups within DoD are invited to submit proposals for services to meet these requirements, either by building them from scratch or by migrating them from legacy components. These requirements must then be analyzed in detail and matched to existing functionality to determine what can be used as is, what has to be modified, and what has to be newly developed.

Such an SOA can impose a number of constraints on organizations that are developing ADSs from existing legacy components. Some of the requirements for developers of ADSs include

1. An ADS must be self-contained; that is, it should be able to be deployed as a single unit.

   As expressed earlier, the reason why SOAs have been a success, especially in industry, is that they provide standard interfaces to legacy systems, while these systems remain largely unchanged.

   This is not the case in the sample target SOA. Services must be independent so that they can be deployed as needed on standardized platforms. In a legacy component, functionality that has been identified as part of a service must be fully extracted from the system, including code that corresponds to shared libraries or the core of a product line.
2. In the target SOA, an ADS may need to be deployed on a specific operating system.

   For legacy components deployed on a different operating system, this could be a problem, especially if there are dependencies on the operating system through direct system calls or if there is a dependency on commercial products that are available only for specific systems. Ideally, system calls should be eliminated. If it is not possible, they should be evaluated to see if there are equivalents in the target operating system or if this functionality is part of one of the common services.
3. All services will share a common data model and all data will be accessed through a Data Store common service.

   The need for a common data model is driven by a desire for information to be shared and understood by all user applications. As a result, services will no longer define internal data. All data will be defined as part of the common data model. Legacy components must replace all dependencies on databases and file systems with calls to the data store service and make sure that all the data they need is part of the common data model.
4. An ADS will use the Discovery common service to find and connect to other services.

   If the ADS will rely on other services, code to discover and connect to these services will have to be written. Once the service is developed, it must be advertised. This is done by registering the service with the naming service. Once this advertised service has been registered, other applications that wish to use this service will perform a discovery on the available services and choose which service(s) they desire to use.
5. An ADS will use the Communications common service for communicating with other services.

   The target SOA provides tools for generating data readers and data writers that will take incoming and outgoing data and format it accordingly.

Information Required for Decision-Making

DoD migrations to SOAs (such as the notional example that we outlined above) will likely rely less on semi-automated migration and more on careful analysis of the feasibility and magnitude of the effort involved. Such an analysis should consider:

1. Requirements from potential service users. It is important to know what

applications would use the services and how they would be used. For example, what is the information expected to be exchanged? In what format?

2. Technical characteristics of the target environment. There are many technical underpinnings that must be understood, especially in proprietary environments, such as bindings, messaging technologies, communication protocols, service description languages, and service discovery mechanisms.
3. The architecture of the legacy system. It is critical to identify architectural elements that could be problematic in the target environment or that could increase the difficulty of the effort, such as dependencies on commercial products or specific operating systems, or poor separation of concerns.
4. The effort involved in writing the service interface. Even if it is expected that the legacy system will remain intact, there must be code that receives the request, translates it into calls to the legacy systems, and produces a response.
5. The effort involved in the translation of data types. Service interfaces usually prescribe a set of data types that can be transmitted in messages. For newer legacy systems and basic data types, this can be a small effort, especially if messages are XML documents. But, in the case of complex data types such as audio, video, and graphics, or in legacy programming languages that do not provide capabilities for building XML documents, this effort can be non-trivial.
6. The effort required to describe the services. In an SOA, services advertise their capabilities for other systems to use, and systems find the services they need by using the discovery mechanism prescribed by the target environment. The more detailed and precise the description of the service, the greater the chances it will be discovered and used appropriately. In critical DoD situations, the description may have to include information about qualities of service, such as performance, reliability, and security, or it may include service level agreements (SLAs) [SLMLC 05].
7. The effort involved in writing service initialization code and operational procedures. Code that is deployed as services will need to initialize itself, announce its availability, and be ready to take requests. This will require the establishment of operational procedures for the deployment of services.
8. Estimates of cost, difficulty, and risk. The information gathered in the previous points should provide for realistic estimates.

Conclusions and Next Steps

To address these issues, it is important to

- gather evidence about potential services that can be created from the legacy components
- gather sufficient detail about the target SOA to support decisions about what services may be appropriate and how they will interact with the architecture

The requirements for how potential services interact with the SOA must be identified. Specific types of requirements may include

- standards and technologies to be employed and relevant guidance documents
- execution platforms, substrates, and middleware
- deployment requirements
- special requirements regarding handling of data and state
- specific quality of service requirements that affect potential services directly and end-to-end quality of service requirements that affect related collections of services

The second article in this series will describe an approach for addressing these issues and evaluating services for migration to an SOA.

References

[Alberts 00]
Alberts, D.; Garstka, J.; and Stein, F. *Network Centric Warfare: Developing and*

Leveraging Information Superiority. 2nd Edition (Revised). Washington, DC: CCRP Publication Series. February 2000.

[Brown 02]
Brown, A; Johnston, S.; and Kelly, K. Using Service-Oriented Architecture and Component-Based Development to Build Web Service Applications. Cupertino, CA: Rational Software Corporation. 2002.

[Lewis 04]
Lewis, Grace and Wrage, Lutz. Approaches to Constructive Interoperability (CMU/SEI-2004-TR-020). Pittsburgh, PA: Software Engineering Institute, 2004.

[SLMLC 05]
The Service Level Management Learning Community.

[Stevens 03]
Stevens, M. "Service-Oriented Architecture Introduction." 2004.

About the Authors

Grace Lewis is a Senior Member of Technical Staff at the Software Engineering Institute (SEI) of Carnegie Mellon University (CMU), where she is currently working in the areas of constructive interoperability, COTS-based systems, modernization of legacy systems, enterprise information systems, and model-driven architecture. Her latest publications include several reports published by Carnegie Mellon on these subjects and a book in the SEI Software Engineering Series. Grace has over fifteen years of experience in Software Engineering. She is also a member of the technical faculty for the Master in Software Engineering program at CMU.

Edwin Morris is a Senior Member of the Technical Staff at the Software Engineering Institute, assigned to the Integration of Software-Intensive Systems (ISIS) Initiative. He is currently investigating approaches to achieving technical interoperability between complex systems and programmatic interoperability between the organizations that build and maintain them. Previous activities involved improving processes and techniques for the evaluation and selection of COTS products, and the development of the COTS Usage Risk Evaluation (CURE) technology. Before coming to the SEI, Ed developed custom operating systems for embedded microprocessors along with support tools to predict and monitor the performance of real time systems.

Dennis Smith is the Lead for the SEI Initiative on the Integration of Software Intensive Systems. This initiative focuses on addressing issues of interoperability and integration in large scale systems and systems of systems. Earlier, he was the technical lead in the effort for migrating legacy systems to product lines. In this role he developed the method “Options Analysis for Reengineering, OARS to support reuse decision-making. Dr. Smith has also been the project leader for the CASE environments project. This project examined the underlying issues of CASE integration, process support for environments and the adoption of technology.

Dr. Smith has published a wide variety of articles and technical reports, and has given talks and keynotes at a number of conferences and workshops. He has an M.A. and PhD from Princeton University, and a B.A from Columbia University.

The views expressed in this article are the author's only and do not represent directly or imply any official position or view of the Software Engineering Institute or Carnegie Mellon University. This article is intended to stimulate further discussion about this topic.

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

Large-Scale Work—Part I: The Organization

NEWS AT SEI

Author

Watts S. Humphrey

This library item is related to the following area(s) of work:

Process Improvement

This article was originally published in News at SEI on: February 1, 2005

With this column, I start a new series of articles about the issues faced by large-scale development projects. Since large-scale development is an enormous subject, I plan to touch only on those topics that I think are particularly interesting or especially important. I currently plan to cover two problems. First, large software projects are almost universally troubled, and second, all large-scale systems-development projects of almost every kind now involve large amounts of software. Unfortunately, this implies that almost all kinds of large-scale development projects will be troubled unless we can devise a better way to develop the software parts of these large-scale systems. The increasing need for large-scale system-development projects raises many questions and presents a significant challenge to those of us in the development business.

Emergent Properties of Systems

Perhaps the greatest single problem with large-scale system development concerns what are called the emergent properties of these systems. These are those properties of the entire system that are not embodied in any of the system's parts. Examples are system security, safety, and performance. While individual components of a system can contribute to safety, security, and performance problems, no component by itself can generally be relied on to make a system safe, secure, or high performing.

The reason that emergent properties are a problem for large-scale systems is related to the way in which we develop these systems. As projects get larger, we structure the overall job into subsystems, then structure the subsystems into products, and refine the products even further into components and possibly even into modules or parts. The objective of this refinement process is to arrive at a series of "bite-sized projects"

Related Links

News

SEI to Co-Sponsor 26th Annual IEEE Software Technology Conference

SEI Cosponsors Agile for Government Summit

Training

See more related courses »

Events

Team Software Process (TSP) Symposium 2014

Nov 3 - 6

that development teams or individual developers can design and develop.

This refinement process can be effective as long as the interfaces among the system's parts are well defined and the parts are sufficiently independent that they can be independently developed. Unfortunately, the nature of emergent properties is that they depend on the cooperative behavior of many, if not all, of a system's parts. This would not be a problem if the system's overall design could completely and precisely specify the properties required of all of the components. For large-scale systems, however, this is rarely possible.

While people have always handled big jobs by breaking them into numerous smaller jobs, this can cause problems when the jobs' parts have interdependencies. System performance, for example, has always been a problem, but we have generally been able to overpower it. That is, the raw power of our technology has often provided the desired performance levels even when the system structure contains many inefficiencies and delays.

As the scale of our systems increases, and the emergent properties become increasingly important, we now face two difficult problems. First, the structural complexity of our large organizations makes the development process less efficient. Since large-scale systems are generally developed by large and complex organizations, and since these large organizations generally distribute large projects across multiple organizational units and locations, these large projects tend also to have complex structures. This added complexity both complicates the work and takes added resources and time.

The second problem is that, as the new set of emergent properties becomes more important, we can no longer rely on technology to overpower the design problem. Security, for example, is not something we can solve with a brute-force design. Security problems often result from subtle combinations of conditions that interact to produce an insecure situation. What is worse, these problems are rarely detectable at the module or part levels.

A War Story

Some years ago, while I still worked at IBM, I was made director of programming. My organization had about 4,000 software professionals in 15 laboratories and 6 countries. While this group was responsible for developing all of the software to support IBM's products, about half of the group was working on one big system: OS/360. These people were highly capable and motivated, but the OS/360 schedule had slipped three times during the past year and a half, and no one believed any of our dates. Since IBM was starting to ship the 360 hardware without the software, the lack of a believable schedule was a major marketing problem.

Marketing argued that the customers would be willing to run their existing system software temporarily with the emulators provided with the 360 systems, but that they would not order many new systems until they had a believable plan for OS/360 software support. We needed a schedule right away, but it had to be one that we would meet without fail. If we had another schedule slip, no one would believe us again.

Before this job, I had run several software projects and also managed some large hardware projects. Therefore, I had learned that it is practically impossible to produce a reliable schedule without first producing a detailed plan for the work. Since the several thousand developers in my group had never before made detailed plans, they didn't know how to do it. What was worse, they didn't believe that planning was important. They knew how to code and test, so that is what they did. Without plans to guide them, the projects were pretty chaotic, and nobody had time to do anything but code and test.

I had to do something to make planning important. So, to get everybody's attention, I established a new operating procedure: Without a detailed plan, no one could announce or ship any software product. I even threatened to cut the budget for any project that did not have a plan. I gave the groups 60 days to produce plans and to review them with me personally. This made planning a crisis. When I got the plans, I reviewed them all and made the developers defend them. We lengthened many of the schedules but never cut a single one. In fact, I even added a 90-day cushion to every

committed date.

This worked. We did not miss a single date for the next two and a half years. Soon, however, we started to eat up my 90-day cushion. We had not appreciated the consequences of a multi-release delivery plan. Every schedule slip for every release had a cumulative effect on every subsequent release. After about two years, these small schedule slips finally added up to my 90-day cushion. However, by then everybody believed our dates, so the subsequent minor schedule adjustment was not a problem. But after that, we no longer committed delivery dates to customers that were more than about a year out. By then, we were meeting all of our delivery dates, and even beating the hardware.

The project-planning procedure worked extremely well. While it imposed a demanding planning discipline on the development groups, they continued producing plans even after I moved on to another job. Unfortunately, with the mechanism we established for plan management, it was easy to add further procedures, so many managers did. Over time, this simple plan-management system became a bureaucratic jungle. The initial planning controls we established had been relatively simple and saved IBM billions of dollars. However, with the added controls, projects could no longer respond quickly to special customer needs, and bureaucracy became a big company problem.

The Tropical Rain Forest

The fundamental problem of scale is illustrated by analogy to the ecological energy balance in a tropical rain forest. In essence, as the forest grows, it develops an increasingly complex structure. As the root system, undergrowth, and canopy grow more complex, it takes an increasing percentage of the ecosystem's available energy just to sustain the jungle's complexity. Finally, when this complexity consumes all of the available energy, growth stops.

The implication for both projects and organizations is that, as they grow, their structure gets progressively more complex, and this increasingly complex structure makes it harder and harder for the developers to do productive work. Finally, at some point, the organization gets so big and so complex that the development groups can no longer get their work done in an orderly, timely, and productive way. Since this is a drastic condition, it is important to understand the mechanisms that cause it.

Organizational Growth

In principle, organizations grow because there is more work to do than the current staff can handle. However, this problem is usually more than just a question of volume. As the scale increases, responsibilities are subdivided and issues that could once be handled informally must be handled by specialized groups. So, in scaling up the organization, we subdivide responsibilities into progressively smaller and less meaningful business elements. Tasks that could once be handled informally by the projects themselves are addressed by specialized staffs. Now, each staff has the sole job of ensuring that each project does this one aspect of its job according to the rules. Furthermore, since each staff's responsibility is far removed from business concerns, normal business-based or marketing-based arguments are rarely effective. The staffs' seemingly arbitrary goals and procedures must either be obeyed or overruled.

This growth process generally happens almost accidentally. A problem comes up, such as a missed schedule, and management decides that future similar problems must be prevented. So they establish a special procedure and group to concentrate on that one problem. In my case, this was a cost-estimating and planning function that required a plan from every project. Each new special procedure and group is like scar tissue and each added bit of scar tissue contributes to the inflexibility of the organization and makes it harder for the developers to do their work. Example staffs are pricing, scheduling, configuration management, system testing, quality assurance, security, and many others.

The Enemy

Since these specialized staffs are all designed to monitor and control the projects, the projects view them as the enemy. So, if they can get away with it, the projects simply ignore the staffs. To guard against this, management establishes a review procedure where the staffs have the authority to sign off at key project milestones. If the project

team does not do its job properly, the staff does not let the project proceed. What is most insidious about this situation is that the staffs are all enforcing rules that management and most objective observers would agree are needed. However, because of the tangle of approvals and sign-offs, the process consumes a great deal of the project's energy. Also, once the staffs have power, they often use that power to push pet objectives that are not strictly in their official charter.

While this review and approval strategy generally guarantees that the projects pay attention to the staffs, the projects must also surmount an increasingly complex bureaucratic tangle of approvals just to do their work. This problem is bad enough when all projects were similar but, as projects get larger and more complex, they, too, become increasingly specialized. They then must use custom-tailored processes and procedures that are almost impossible to establish in a large and complex organizational jungle. Because of the energy consumed in getting bureaucratic approvals, the projects generally have to follow a process that doesn't precisely fit their needs. Now, just like a tropical rain forest, an increasing proportion of the organization's resources are devoted to delaying or stopping projects. Because of the enormous energy required to fight the bureaucracy, the projects have progressively less energy left to design and build their products. That is when rapid growth stops.

While there is no magic solution to the problems of project scale, the situation is not hopeless, and there are useful steps we can take. Subsequent columns outline the most promising avenues for addressing these problems. I will also outline some principles to consider when working on large-scale development projects, some of the consequences of scale-related development problems, and some strategies for solving them.

Acknowledgements

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

In closing, an invitation to readers

In this particular series of columns, I discuss some of the development issues related to large-scale projects. Since this is an enormous subject, I cannot hope to be comprehensive but I do want to address the issues you feel strongly about. So, if there are aspects of this subject that you feel are particularly important and would like covered, please drop me a note with your comments, questions, or suggestions. Better yet, include a war story or brief anecdote that illustrates your ideas. I will read your notes and consider them when planning future columns.

Thanks for your attention and please stay tuned in.

Watts S. Humphrey
watts@sei.cmu.edu

About the Author

Watts S. Humphrey founded the Software Process Program at the SEI. He is a fellow of the institute and is a research scientist on its staff. From 1959 to 1986, he was associated with IBM Corporation, where he was director of programming quality and process. His publications include many technical papers and several books. His most recent books are *Introduction to the Team Software Process* (2000) and *Winning With Software: An Executive Strategy* (2002). He holds five U.S. patents. He is a member of the Association for Computing Machinery, a fellow of the Institute for Electrical and Electronics Engineers, and a past member of the Malcolm Baldrige National Quality Award Board of Examiners. He holds a BS in physics from the University of Chicago, an MS in physics from the Illinois Institute of Technology, and an MBA from the University of Chicago.

Find Us Here

Share This Page

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

Using Product Line Analysis to Get Started With Software Product Lines

NEWS AT SEI

Author

Patrick Donohoe

This library item is related to the following area(s) of work:

Software Product Lines

This article was originally published in News at SEI on: March 1, 2005

Suppose your organization has decided to pursue a software product line approach. There seems to be a sound initial business case for doing so, the set of likely products in the product line has been identified, there is a market for such a product line, and your organization believes it has the domain expertise and technical savvy to make it all work. Now what? How do you combine these disparate sources of information into a coherent set of preliminary requirements that will shape the product line architecture and production strategy?

Product line analysis (PLA) is an initial and relatively brief pass at requirements engineering for a product line of software-intensive systems. Its goal is to identify opportunities for large-grained reuse across the product line. PLA is the link between the recognition of a business opportunity and the design of a product line architecture. It incorporates the views of multiple product line stakeholders in a preliminary requirements model that includes the functional features of products and the software quality attributes (e.g., performance, modifiability) of both the products and their development. The stakeholders providing the necessary input include marketers, managers, customer representatives, and architects. The requirements model created by PLA identifies common requirements across the product line and their allowed variants.

The primary benefit of PLA is its early identification of the major issues affecting a product line and its development (e.g., how much of the functionality and quality attributes could be implemented in core assets and how much will need to be done by

Related Links

Training

Software Product Lines - eLearning

Software Product Lines

See more related courses »

product developers, and the technical and business consequences of the proposed reuse strategy). PLA is based on a systematic modeling process that incorporates the organization's business goals and constraints, clarifies and refines assumptions about the product line scope, and provides early information about the technical feasibility of the product line. PLA also provides input to the decisions and tradeoffs concerning allocation of resources to core asset and product development. PLA helps managers identify risks in their approach to adopting software product lines. It also helps asset developers (including architects) who need usable AND useful representations of requirements, not a set of "shelfware" binders.

What the SEI now calls "product line analysis" grew out of its early domain-analysis and requirements-modeling work with industrial customers building software product lines. Because product lines span multiple domains and have multiple stakeholders, domain-analysis techniques alone did not provide a complete solution. As a consequence, the SEI approach uses techniques from domain analysis, but also borrows from work in object technology, use-case modeling, and architecture definition and evaluation.

A joint SEI-customer team used an early version of PLA to define the requirements for a product line of automotive software. The requirements model captured common requirements and variation points and allowed early exploration of the product line architecture to proceed in parallel with detailed requirements engineering.

What PLA Can Do for Your Organization

Because PLA is focused on large-grained reuse across a product line, it does not address all the requirements. It is an iterative, incremental process of eliciting, analyzing, specifying, and verifying the early requirements for a product line based on an initial business case and market analysis. The output of the process is a requirements model comprising four interrelated work products. The work products are based on object modeling, use-case modeling, and feature-modeling techniques:

- The *use-case model* specifies the product line stakeholders and their key interactions with the product line. These stakeholders will verify the acceptability of the product line (and of the requirements).
- The *feature model* specifies the stakeholders' views of the product line. It captures the functional features of products and the software quality attributes of the product line and its products.
- The *object model* specifies the product line responsibilities that support those features and the commonality and variability of the responsibilities.
- The *dictionary* defines the terminology used in the work products and supports a consistent view of the product line requirements.

Together these work products form the basis of a systematic method for capturing and refining requirements for current products, future envisioned products, stakeholder needs and expectations, and associated rationales and tradeoffs. The work products are first populated with an initial set of use cases, features, responsibilities, and terminology, and then refined iteratively and incrementally. Use cases and features determine the elicitation of the product line requirements. Features and objects are analyzed for commonalities and variabilities, consistency, quality, interactions, and priority. The product line stakeholders verify the accuracy and completeness of the resultant requirements model. The modeling stops when opportunities for large-grained reuse can no longer be identified.

A software product line succeeds because the commonalities shared by the products can be exploited to achieve economies of production. PLA mitigates the risk of product line adoption by combining information from multiple product line stakeholders in a form that permits reasoning about the allocation of functional features and quality attributes to assets and products. The premise of PLA is that a sound initial understanding of the problem to be solved is essential before an organization embarks on a software product line as a solution. The model of a product line that emerges from PLA also provides early identification of the architecturally-significant requirements. The modeling process incorporates the organization's business goals and constraints, clarifies and refines assumptions about the product line scope, and

provides an early indication of the technical feasibility of the product line.

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

Writing the Book on Process Improvement: An Interview with Watts Humphrey

NEWS AT SEI

Author

Susan Kushner

This library item is related to the following area(s) of work:

Process Improvement

This article was originally published in News at SEI on: March 1, 2005

Watts Humphrey founded the Software Process Program of the Software Engineering Institute (SEI) at Carnegie Mellon University. During a formal ceremony at the White House in the spring of 2005, the President of the United States awarded him the National Medal of Technology for his contributions to the field of software engineering. news@sei *recently talked to Watts about his experience at the White House, his latest book, and his upcoming publications.*

Watts, thank you for taking the time to talk to news@sei. Congratulations on being awarded the National Medal of Technology. March must have been an exciting time in your career.

It certainly was. I really appreciate all of the comments, support, and the marvelous reaction of everyone that's been involved. It's very nice, and I appreciate the help of all the SEI folks.

I followed the media coverage of your visit to the White House, but it is difficult imagine what the experience was truly like. Can you tell us more about it from your perspective?

It took a while to sink in. It started to when I talked to the contact there about the arrangements. Everything was scheduled down to the minute. Through all the mechanics involved, I began to realize this is a fairly significant operation. On the bus from the hotel to the White House, I sat next to a gentleman who was also getting an award. He'd won a Nobel Prize. I started to realize that this wasn't just an ordinary

Related Links

News

SEI to Co-Sponsor 26th Annual IEEE Software Technology Conference

SEI Cosponsors Agile for Government Summit

Training

See more related courses »

Events

Team Software Process (TSP) Symposium 2014

Nov 3 - 6

crowd of folks.

We arrived, were greeted, and entered the White House. A whole section on the first floor was open for this event, including the Blue Room and the East Room. We were escorted to the Blue Room where President Bush actually stopped and shook each of our hands and talked with us. I was very impressed with his presence because when he stopped to talk to me, it was clear that he was talking to me. It was kind of surprising. With a large group, it's difficult to shake everyone's hand and say hello, and to focus on each individual. He's obviously a pro at that.

The occasion went very smoothly. It was delightful to have almost all of my family there. After the event, we had a reception and a meal. It was beautifully done.

Also in attendance were some other high-profile people such as Secretary of Commerce Carlos Gutierrez, Under Secretary of Commerce for Technology Phillip Bond, and several former Medal of Honor laureates. Was there an opportunity to interact with any of them?

Yes, I did talk to quite a few people. I chatted briefly with Phil Bond and talked to a number of the award recipients. There were two Nobel Prize winners in the group and I chatted with one of them at length about his work in the field of cancer research.

Were there any surprises during the visit?

I was surprised that it all went so smoothly. One piece of work they did was very impressive. A group was hired to put together a two- to three-minute video about each of the 14 laureates, which was shown at the reception on Monday night. They took bits and pieces, a picture or two, a video clip from the SEI, and one from IBM and put together a very coherent story.

What is the one moment that particularly stands out as the defining moment of this experience?

Essentially, after the President placed the medal on me. Then we turned out to face the audience for a picture as we'd been told to do. It was then I realized, wow, it really did happen!

March was a busy month. You visited the White House and had your ninth book, PSP: A Self-Improvement Process for Software Engineers, published. How is this book different from your first book on the PSP, A Discipline for Software Engineering?

When I wrote the first book, I had used the PSP myself and had data from one class. But I wrote the book before anyone else had used the PSP. Then I used the manuscript to teach a course at Carnegie Mellon, and I had two other people use it to each teach a course. I held off on publishing the book until I was confident that the PSP would work. At that time, I had data on about two dozen people.

The data set I used when writing the new PSP book was from 8000 programs using the PSP. Any time I had a question, or if I made an assertion about design methods, quality, or planning, I'd look at the data to make sure I was right. Using an enormous

amount of data to support what you're doing gives it a great deal more authority.

I cut the size of book down significantly. It's only about half as long as the previous version. I cleaned it up and responded to suggestions from readers. I tried to deal with questions up front and make it more modern and accessible to the current audience. This industry is changing so quickly that a book written 10 years ago can't cover everything that is current today.

You set aside time each day specifically to write. Are you working on a new book?

I actually have two books now under contract with Addison-Wesley. I drafted both books a couple of years ago, and we've been using them in courses. The manuscript I'm finishing now is a Team Software Process (TSP) book about leading a development team. The team leader is probably the most important person on the team. The leader's behavior and ability to guide, support, and motivate the team is the most important factor in team success.

The other book I'm working on is about coaching development teams. Earlier, I had thought of leading and coaching as similar roles, but I realize now that they're quite different.

So you're planning to further define each of these roles in its own book?

Yes, the need for coaches on development teams hasn't been sufficiently emphasized. We all know how vital coaches are on athletic teams. When a football team or a baseball team isn't doing well, often one of the first things management considers is changing the coach. The coach is important in motivating people on teams.

The coach role in the TSP is interesting because coaches are not part of the team. They support the team, but they're independent of it. They focus on building the team and supporting individuals on the team. Even though they support each other, the coach role is different from the team-leader role. Team leaders build a product and use the team to do it. Coaches build teams and use the product effort to do it.

The PSP and TSP could be applied to many endeavors, not just software development. Is that an accurate statement?

I use PSP for writing books and preparing talks and presentations. For seven of the books I've published with Addison-Wesley, I haven't yet missed a date. We waste a lot of time frittering around thinking about how to proceed. Using a procedure and a process can help you work much more efficiently. It's helpful to work against a plan, and it's rewarding to know that I'll get the book done on time. I'll almost certainly run into something that I hadn't expected along the way, but that's normal. When you know where you stand, you have room to handle the surprises.

That's really the big picture I'm trying to get across. The PSP and the TSP are a way to build a rewarding and enjoyable life. Many people view processes only as constraining disciplines not as the enablers that they really are.

CMMI V1.2: What's Changing?

NEWS AT SEI

Author

Mike Phillips

This library item is related to the following area(s) of work:

CMMI

Process Improvement

This article was originally published in News at SEI on: March 1, 2005

My last column focused on appraisals, and now is a good time to update you on where we see ourselves going with the CMMI Product Suite. This is the first in a series of columns to provide you a broad understanding of what will be included-and what will not be included-in V1.2.

In this first segment, let me first review our process. In our preparation for V1.2, we used the change request (CR) process to identify the potential areas for improvement. For major decisions, we provided decision papers to the CMMI Steering Group to get strategic direction. We then created change packages with more detailed proposed conceptual solutions, always tied to CRs we received, that required review and approval by our multi-organizational Configuration Control Board (CCB). All of the proposed changes have now been approved, and we have begun the task of making changes to the CMMI Product Suite documents. Once these changes are redlined, we will return implementation packages containing these changes to the CCB to get its approval for each of the process areas.

What architectural changes will be in V1.2?

As we began our investigation of improvement possibilities, we determined that we needed to be able to accommodate expanded enterprise coverage without continually increasing the size of the models that organizations need to guide their part of the work. While we could easily imagine gathering best-practice inputs to guide process improvement across more of the life cycle and into new domains, we needed to examine our architectural approach. One element of this review will be seen in V1.2, as

Please note that current and future CMMI research, training, and information has been transitioned to the CMMI Institute, a wholly-owned subsidiary of Carnegie Mellon University.

Related Links

News

SEI to Co-Sponsor 26th Annual IEEE Software Technology Conference

SEI Cosponsors Agile for Government Summit

Training

See more related courses »

Events

Team Software Process (TSP) Symposium 2014

Nov 3 - 6

it helps to clarify the current material.

In the existing models, we describe the total collection of best practices for product development as "CMMI-SE/SW/IPPD/SS." We described each of the bodies of knowledge comprising existing models as "disciplines." While systems engineering (SE) and software engineering (SW) are bodies of knowledge that can stand alone or with other bodies of knowledge, integrated process and product development (IPPD) and supplier sourcing (SS) are somewhat different and must be combined with SE, SW, or both. So with V1.2, the full development collection will be described as "CMMI-SE/SW+IPPD+SS." We believe this new way of naming CMMI models will also make the differences between models within the development collection easier to recognize. Organizations choosing to address the engineering disciplines all use the same set of practices. Those who wish to add IPPD or SS coverage add practices and goals to cover these environments.

This architectural change is a small one for V1.2, but it has larger implications because it helps position the CMMI model framework for future expansion. We are calling these future collections of models "constellations." The current collection of CMMI models comprises the development constellation. As the product suite expands to new constellations, we will seek to maximize the commonality among all future and current constellations to allow the generation of multiple model variations.

The first expansion beyond the development constellation was already approved by the CMMI Steering Group. This new constellation will address services. The release of the service constellation will occur after the release of V1.2, so I will not discuss the details here. Suffice it to say that this is the first approved expansion, and its release will occur a few months after the release of V1.2. At that point, we will have a "CMMI-Development" family of models and a "CMMI-Services" set. We envision that most of the process areas will be identical for both collections, but some process areas will be unique to Development and others to Services. The initial investigation suggests that the Process Management, Project Management, and Support process areas might be common across both collections, with only the current Engineering group being the place for selected process area replacements for CMMI-Services.

Are there any expansions planned for V1.2?

One expansion that the Steering Group has approved is actually a clarification-coverage of the hardware discipline. In some organizations, hardware development centers are not being included within the scope of CMMI-based process improvement because of perceptions that the models are "just for systems engineering and software engineering." We are planning to add informative material in amplifications, typical work products, and examples that apply specifically to those engineers producing the hardware elements of software-intensive systems. We found that these additions better covered product development as a whole and improved the recognition that development efforts must plan appropriately for production.

Another expansion we will include in V1.2 is a broadening of CMMI coverage of the development environment. Currently, the Organizational Environment for Integration (OEI) process area provides some coverage of these processes, and other process areas provide some additional information. However, one of the source models for CMMI, EIA 731, contains additional practices covering the engineering environment [EIA 98]. Furthermore, CRs have indicated that coverage of the work environment is desirable to address safety and security. A few provocative CRs have suggested that we need to include some of the best practices associated with "business continuity," particularly after experiences or organizational shocks such as the September 11 disaster. We determined that the kind of expanded coverage we provide for IPPD in the Integrated Project Management (IPM) process area was a good model for the approach. A potential new process area, "Work Environment," (WE) has been proposed to include the OEI practices and goals as IPPD additions. I'll discuss more about the final approach we take in my column in the next issue.

Are there any consolidations or other ways to reduce the complexity?

First, we think that presenting both the staged and continuous representations in one document reduces one aspect of model complexity. The dual-representation book version of CMMI [Chrissis 03] contains the same total number of pages as a previous

model that includes only one representation. We are pleased with this outcome.

We are removing two components from CMMI models: advanced practices and common features. These are legacy artifacts from two CMMI source models that complicate the depiction of the material and no longer serve the purpose they once did in their sources. Advanced practices are artifacts inherited from the EIA 731 source model, and common features are artifacts inherited from the SW-CMM source model [SEI 95]. While each of these components has had value in V1.0 and V1.1, the need to use them in the continuing depiction of the CMMI model (2006 and beyond) appears marginal and outweighed by the potential for simplification. Generic practices will continue as the main measure of capability levels but will not be characterized by their common-feature identifier.

In addition to simplifying the numbering scheme, several practices that were base practices in the continuous representation will no longer be in the document. Instead, the base and advanced pairing of practices will be replaced by the advanced practice only. For example, we will describe only "eliciting" requirements, rather than seeking to differentiate (particularly in appraisals) whether requirements have been "gathered" or "elicited."

We investigated combining the two process areas that deal with suppliers-Supplier Agreement Management (SAM) and Integrated Supplier Management (ISM)-but concluded that ISM needed to be addressed in a distinct, proactive manner that deserved separate level-three coverage.

A related clarification is to move the commercial off-the-shelf (COTS) choices from their current location within SAM to a related section within Technical Solution (TS). Many CRs noted that the effort to "select product-component solutions" was a better place to frame the discussion about COTS than its current location in SAM.

With increasing numbers of organizations exploring high-maturity practices, we have been encouraged by CRs to give clearer guidance on what it means to apply the practices summarized in the level four and five generic practices to the various process areas. So we will provide more elaborations in these areas for V1.2.

Summary

As you can see, the major purpose of our work for V1.2 is to clarify and simplify, with only limited expansion of coverage. We do not envision the need for anyone who has taken CMMI courses to return for revised training. We do recognize that the changes will require some familiarization and perhaps some minor adjustment to mappings or practice implementation indicator documents (PIIDs). Therefore we will establish a period of continued support for V1.1 appraisals to minimize any disruption with the change.

In the next quarterly column, I'll be able to provide more detail on the changes for V1.2, as we prepare to pilot the proposed update.

References

[Chrissis 03]
Chrissis, Mary Beth; Konrad, Mike; and Shrum, Sandy. CMMI: Guidelines for Process Integration and Product Improvement. Boston, MA: Addison-Wesley, 2003.

[EIA 98]
Electronic Industries Alliance. Systems Engineering Capability Model (EIA/IS- 731). Washington, DC, 1998.

[SEI 95]
Software Engineering Institute. *The Capability Maturity Model: Guidelines for Improving the Software Process*. Reading, MA: Addison-Wesley, 1995.

About the Author

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

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

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

Shifting Perspective to Achieve and Sustain Enterprise Security

NEWS AT SEI

Author

Julia H. Allen

This article was originally published in News at SEI on: March 1, 2005

Security-conscious leaders ensure that they are adequately and accurately informed with respect to risk management, business continuity, and organizational resilience, all of which affect security-governance actions. In our research on managing for enterprise security, we discuss the necessity of a shift in perspective,[1] point of view, or frame of reference to be in a position to ask the right questions, as follows:

> Security lives in an organizational and operational context, not as an isolated discipline. Effective security must take into account the dynamically changing risk environment within which most organizations are expected to survive and thrive. To achieve and sustain an adequate level of security that directly supports the mission of the organization, leaders must shift their point of view (or frame of reference) and that of their organization from an information-technology-based, security-centric, technology-solution perspective to an enterprise-based, risk management, organizational continuity and resilience perspective. This requires moving well beyond ad-hoc, reactive approaches to security (lacking process and procedure, and dependent upon individual heroics) to approaches that are process centered, strategic, and adaptive. The CSO [and CISO] must be able to draw upon the capabilities of the entire organization so that they can be deployed to address a problem requiring an enterprise-wide solution set. However, because security isn't a one-shot activity, it also means being able to achieve it in a way that is sustainable—systematic, documented, repeatable, optimized, and adequate with respect to the organization's strategic drivers. [Caralli 04]

(< 5 minute) survey.

The presence of this shift in perspective increases the likelihood of involving the right stakeholders and obtaining the right information required to make well-informed governance decisions about security oversight, investment, and performance. The shifts most applicable to governance are briefly summarized below and in the table below. They are covered in greater detail in our technical report *Managing for Enterprise Security*. [Caralli 04]

- Scope: From viewing security as a technical or technology-centric problem to viewing security as an enterprise-management problem. Scope answers the question, "What is the scope and extent of security concern within the enterprise? "
- Ownership: From security ownership by those with technical expertise to ownership by the business, which is the driver and ultimate benefactor. Ownership answers the questions "Who has the authority to act?" and "Who is accountable and responsible?"
- Focus: From an intermittent focus on security when something bad happens to treating security as an accepted and expected business process and an included cost of doing business. Focus answers the question "How is security considered with respect to other fundamental enterprise operating principles?"
- Funding: From treatment as a discretionary expense, burden, or tax to treatment as an expense and investment for the business projects and processes that security supports. Funding answers the questions "How does the organization fund the sustainment of adequate security?" and "How is security return on investment (ROI) calculated?"
- Goal: From leaders asking the question, "Are we secure?" to leaders asking the more useful and relevant question, "With respect to security, have we taken sufficient steps to ensure that the business and its critical assets are adequately protected and properly resilient?"

From	**To**
Scope: Security is a technical problem: • technical network (hardware, software, infrastructure) • technical requirements (protect the perimeter) • technical assets (desktops, laptops, servers, databases) • technical specialty (in the realm of IT and system administrators)	Security is an enterprise-wide problem: • enterprise network (people, processes, business units) • enterprise requirements (privacy, asset protection) • enterprise assets (customer data, employee data, communication) • enterprise core competency
Ownership: Security has a technical owner: • IT is the driver, owner, and primary benefactor. • Technical personnel are responsible for security. • The CSO/CISO is considered a technical adviser.	Security is owned by the enterprise: • The enterprise is the driver, owner, and primary benefactor. • Business leaders understand security and have security responsibilities. • The CSO/CISO is considered an adviser to the business.

	• All employees understand their responsibilities with respect to security.
Focus: There is an intermittent focus on security: • Security is sporadically singled out for attention, investment, and justification. • Risk assessment is applied to security as a special case. • Security is on the agenda to comply with regulatory requirements.	Security is integrated: • Security is a requirement of conducting business, considered in normal planning and business-conduct cycles. • A more secure state results from effective risk-management capabilities. • Existing security controls meet compliance requirements.
Funding: Security is an expense: • The benefit of security is not measured or is hard to measure. • Return on security investments is not required or quantifiable.	Security is an investment: • The benefits of security are measurable, measured, and regularly reported. • Return on security investment is required and quantifiable in business terms. • Security expense and investment is part of all applicable business projects and processes.
Goal: The goal is security: • The focus of security efforts is on threat, vulnerability, and protection. • There is no articulated, desired security state. • There is a potentially excessive deployment of security technologies undertaken in a piecemeal approach.	The goal is business continuity and ultimately resiliency: • The focus of security efforts is on impact, organizational continuity, and preservation of trust. • Adequate security that meets business objectives is the desired state. • Security costs, benefits, and risks are in balance.

1 Earlier work on shifts in perspective from security to survivability, including questions to ask to initiate each shift, can be found in the article “Information Survivability: Required Shifts in Perspective” [Allen 02].

References

[Allen 02]
Allen, Julia; Sledge, Carol. "Information Survivability: Required Shifts in Perspective." *CrossTalk*, July 2002.

[Caralli 04]
Caralli, Richard; Wilson, William. "The Challenges of Security Management." Carnegie Mellon University, Software Engineering Institute, July 2004.

About the Author

Julia Allen is a senior member of the technical staff within the Networked Systems Survivability Program at the Software Engineering Institute (SEI), a unit of Carnegie Mellon University in Pittsburgh, PA. The CERT Coordination Center is also a part of this program.

Allen is engaged in developing and transitioning enterprise security frameworks and executive outreach programs in enterprise security and governance. Prior to this technical assignment, Allen served as acting Director of the SEI for an interim period of 6 months as well as Deputy Director/Chief Operating Officer for 3 years. Her degrees include a B. Sci. in Computer Science (University of Michigan) and an MS in Electrical Engineering (University of Southern California). She is the author of *The CERT Guide to System and Network Security Practices* (Addison-Wesley, June 2001).

The views expressed in this article are the author's only and do not represent directly or imply any official position or view of the Software Engineering Institute or Carnegie Mellon University. This article is intended to stimulate further discussion about this topic.

The SEI Partner Network and the Code of Professional Conduct

NEWS AT SEI

This article was originally published in News at SEI on: March 1, 2005

The SEI has long been a source of important innovations in software engineering. As they mature, these innovations can take the form of training courses or assessment services, but in order for these services to have an impact, they must be transitioned to the users who would benefit from them. To help get the word out, the SEI counts on the SEI Partner Network, an international group of organizations and individuals that are selected, licensed and trained by the SEI to provide leadership in transitioning mature SEI-branded services, including courses, into practice. Through this global network of SEI Partners, SEI courses and services are available to a wider audience than can be served directly by the SEI. “In addition, SEI’s branding and quality activities of the SEI Partner Network help to distinguish SEI Partners from others offering similar courses and services,” says Richard Cox, manager of the Licensing Program at the SEI. “These SEI Partners and SEI-Authorized and Certified Professionals are key to disseminating SEI-branded services, and the software engineering community understands the value of working with SEI Partners to obtain SEI services.”

Growth of the Partner Network

When the SEI licensing program was initially established and only ten organizations were partners, quality-related practices and principles applicable to the delivery of SEI-licensed services were not explicitly articulated. Rather, they were assumptions that were understood by these partners, who knew SEI business culture and worked very closely with the SEI. But a rapid increase in the number of partners has created a need for a more defined quality system with continuous process improvement. (As of May 2005, the SEI Partner Network has grown to over 235 partners.) “The trend now is toward global infusion in many business cultures with different assumptions and pressures and so organic processes must be designed and policies must be normalized rather than exception-based; and responses must be proactive, consistent and clear rather than reactive, inconsistent and fuzzy,” Cox suggests. “In short, the rules of engagement must be explicit rather than driven by unspoken assumptions.”

Building Quality: The Code of Professional Conduct

The SEI began building a quality system for the delivery of SEI-branded services in

2003. The first step that the SEI took was to draft and publish a Code of Professional Conduct for SEI Services. The SEI developed the Code with input from the community of SEI-Authorized and SEI-Certified Professionals and SEI Partners. John Maher, of Organization and Process Improvement Incorporated, led the effort. “Instead of simply adding requirements to the partner agreement, the SEI took a different approach,” Maher says. “They sought contributions from partners and tried to build consensus right from the start on the requirements of the Code.” The Code was first reviewed publicly in November 2003 and went through multiple iterations based on feedback from inside the SEI, SEI Partners and SEI-Authorized Professionals. All submitted change requests were posted publicly and reviewed by the SEI; responses to comments were then posted with each new version. Over 500 people commented on these early drafts.

The Code is designed to protect the reputation of the SEI, SEI Partners and SEI-Authorized and Certified Professionals. “The Code emphasizes the core values of the SEI Partner Network—integrity, excellence and impact—by setting a standard of behavior across the community for the delivery of SEI-branded services” says Cox. “It has become a part of the contractual relationship for authorization, certification and licenses and all SEI Partners and Authorized and Certified Professionals must commit by signature to the Code. The Code applies to all members of the community. In addition, the Code establishes some of the requirements for both a more robust service support system and quality system at the SEI.”

The Code, driven in part by the community of SEI Partners and SEI-Authorized/Certified Professionals defines the behaviors expected from this community when providing SEI-branded services. These behaviors further distinguish the providers of SEI-branded services from those delivering non-SEI services.

Commitments to the Code of Professional Conduct

As of June 1, 2005, 237 organizations and 745 individuals have committed to the Code. The license agreements for those organizations that elected not to commit or did not respond were cancelled for convenience, and the authorizations of those individuals who elected not to commit or did not respond are being revoked. Those organizations and individuals who have their licenses cancelled or authorizations revoked will no longer be permitted to deliver the associated SEI-branded services.

In order to assure that the Code is recognized both in description and practice as fairly representing all parties when violations to the Code have been investigated by the SEI, the Code provides for an elected Review Board for the Code of Professional Conduct for SEI Services (Review Board) from the community of professionals covered by the Code.
The Review Board provides an independent review of the data presented and any conclusions drawn, and its authority is limited to making a recommendation to the SEI Director. The Board (1) considers requests for review of Code investigations and conclusions; (2) conducts independent reviews, when appropriate; and (3) makes recommendations to the SEI Director based on these reviews. The Board may review all aspects of a request, including the data, the process and the conclusion, subject to confidentiality agreements. Maher adds "The Code, and the independent Review Board, help to maintain a level playing field and set ground rules that all partners and certified or authorized individuals have agreed to live by, regardless of organization or location."

The Future of the Code of Professional Conduct

Immediately upon completion of Version 1 of the Code, work began on developing the governance and guidance documents for the Review Board and the required infrastructure to support it. This work was completed in October 2004. As of this writing, the election process is being planned and then nominations will be solicited. Cox adds “As partners use the Code, we’ll get feedback about how it works in practice, and this will trigger changes for Version 2.”

Share This Page

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

Meeting the Challenge of Ultra-Large-Scale (ULS) Systems

NEWS AT SEI

Author

Bill Pollak

This library item is related to the following area(s) of work:

Ultra-Large-Scale Systems

This article was originally published in News at SEI on: April 1, 2005

> *"Given the issues with today's software engineering, how can we build systems of the future that are likely to have billions of lines of code?"*

Claude M. Bolton, Jr., assistant secretary of the Army (Acquisition, Logistics, and Technology), posed this question to the SEI in 2004. The question led the SEI to conduct a research study in ultra-large-scale (ULS) systems: systems that exceed some critical limit of today's software engineering technology. Although Assistant Secretary Bolton initially defined the challenge of ULS systems in terms of billions of lines of code, size is only one limit. Others include unboundedness, continuous requirements evolution, and continuous operation.

The intended outcome of the research study is the creation of a technology roadmap for ULS systems and a collaborative research network that works toward solving the ULS problem. "Our soldiers depend on software and will depend more on software in the future," says Assistant Secretary Bolton. "The Army's success depends on software and the software industry. We need to build reliable ULS systems to meet the needs of the military and society at large."

Software, says Bolton, is the chief enabler of an Army transformation that emphasizes information superiority. "Software makes possible increased situational awareness by providing sensors into networks that allow commanders and soldiers to see first, act

first, and act decisively," he says. But the Army's demands for software are rapidly outpacing its ability to manage software acquisition. The Army's Future Combat Systems (FCS), for which it plans to develop and acquire more than 33 million lines of code (LOC), is already challenging current software engineering capabilities. "We need better tools to meet future challenges," says Bolton, "and neither industry nor government is working on how to do things light-years faster and cheaper. How can future systems be built reliably if we can't even get today's systems right?"

Peter Freeman, assistant director of the National Science Foundation for Computer and Information Science and Engineering, concurs. "Our concepts for new systems seem larger than the reality of developing them," he says. The problem, says Freeman, is that the ideas and concepts needed to meet future challenges may be outside the discipline of software engineering as it is currently defined and understood. "We as a field need to think deeply and carefully to characterize the nature of the problem and the nature of the material with which we work—what is software, what are systems, and what are components? How do we characterize the relationships between these, and how do we characterize the dynamic behavior of systems? We don't have the intellectual basis in place for science and design to do the things that Assistant Secretary Bolton envisions. So we need new ideas."

The study that the SEI is conducting is intended to generate such new ideas. It brings together software experts with a wide range of knowledge and expertise representing the SEI, Carnegie Mellon University, and several other institutions and organizations (see sidebar). The initial intended outcome of the study is a program definition including

- a technology roadmap
- defined areas of investigation
- key organizations to be involved
- funding required

The team agreed that incremental improvement of current approaches will not suffice to solve the ULS system problem. "If our ways of thinking are behind the problems that we face today," says team member Jack Whalen of Palo Alto Research Center, "we need to seriously rethink how we conceptualize the things that we take for granted."

Linda Northrop, director of the SEI Product Line Systems Program and leader of the ULS project at the SEI, agrees. "This is precisely what we've been charged to do," she says. "This is the kind of thinking that Mr. Bolton wants to encourage. To make significant progress in the size and complexity of systems that can be built and deployed successfully, we require a culture shift. Our task is to identify the kinds of research that will effect such a culture shift."

The team held its first meeting on August 16-19 at the SEI's main office in Pittsburgh. At the meeting, participants discussed and identified the organizing elements for a ULS technology roadmap: ULS system characteristics, technical challenge areas, and breakthrough research areas.

"One of the questions we struggled with when we first started is how to define ULS system characteristics," says Mark Klein, an SEI member of the team. "We rejected the simple notion that 'ultra' applied only to the number of lines of code. Somehow as the system grows along various dimensions, conventional wisdom no longer applies. These systems must always run. They are seemingly ubiquitous. They will never be error free. No small number of operators can control them, and the number of operators required would require too much coordination. The distinction between runtime and design is blurred. These are examples of what we mean by 'ULS system characteristics'."

Technical challenge areas are a group of technical capabilities that seem likely to be difficult for ULS systems.

Breakthrough research areas are those that are likely to make the development of ULS systems more tractable—either a new technical approach or a breakthrough in an existing approach needed to address one or more of the ULS technical challenge areas. Three research-study subteams formed at the meeting are currently working on

defining breakthrough research areas related to the defined technical challenges.

The next meeting of the team will be held Nov 29-30, again in Pittsburgh. The SEI plans to complete the effort on April 30, 2006, when a full report will be available.

Invited speakers at the August 2005 ULS System Meeting meeting included

- Assistant Secretary Claude M. Bolton, (on video)
- Peter Freeman, National Science Foundation
- David Emery, DSCI
- Bruce Krogh, School of Electrical and Computer Engineering, Carnegie Mellon University

Members of the external expert panel are

- Gregory Abowd, Georgia Institute of Technology
- Carliss Baldwin, Harvard Business School
- Robert Balzer, Teknowledge Corporation
- Richard Gabriel, Sun Microsystems, Inc.
- Gregor Kiczales, University of British Columbia
- John Lehoczky, Carnegie Mellon University
- Ali Mili, New Jersey Institute of Technology
- Peter Neumann, SRI International
- Douglas Schmidt, Vanderbilt University
- Mary Shaw, Carnegie Mellon University
- Daniel Siewiorek, Carnegie Mellon University
- Kevin Sullivan, University of Virginia
- Jack Whalen, Palo Alto Research Center

SEI members of the ULS team are

- Linda Northrop
- Peter Feiler
- John Goodenough
- Rick Kazman
- Mark Klein

- Rick Linger
- Tom Longstaff
- Jim Over
- Mark Pleszkoch
- Kurt Wallnau

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

Secure Coding in C and C++: C-Style Strings

NEWS AT SEI

Author

Robert C. Seacord

This article was originally published in News at SEI on: April 1, 2005

C++ creator Bjarne Stroustrup has commented, "C makes it easy to shoot yourself in the foot; C++ makes it harder, but when you do it blows your whole leg off." So are programmers still embracing the C and C++ programming languages? The answer is a resounding "yes"—estimates are that C and C++ continue to have more than 3 million users. But using C and C++ creates challenges for programmers.

"The C programming language is a flexible, portable, high-level language that has been used extensively for more than 30 years but is the bane of the security community," says Robert C. Seacord, senior vulnerability analyst at the SEI's CERT Coordination Center (CERT/CC).

Seacord's latest work, *Secure Coding in C and C++* (Addison-Wesley, 2005), is becoming well known in software development circles. Software Developer Magazine editor Rick Wayne recently wrote that "...Seacord dissects the way worms, viruses and other despicable fauna wreak their havoc on unwary programs, and shows how to stop problems at their source (which is the source, so to speak). String buffers are, sadly, still a rich vein for attackers to mine, but Seacord doesn't stop with the obvious, covering pointers, memory management and even unlikely-seeming avenues of attack like the humble integer. In each case, he covers "mitigation strategies" in depth, examining coding practices as well as tools, with plenty of source-code examples."

LinuxWorld Magazine also ran an interview with Seacord as one of its cover features in the November 2005 issue. In the interview, Seacord detailed some best practices for secure coding in C and C++ and addressed some Linux-specific security issues.

Commonly exploited software vulnerabilities are usually caused by avoidable software defects. Seacord's book systematically identifies the program errors most likely to lead to security breaches, shows how they can be exploited, reviews the potential consequences, and presents secure alternatives. Secure Coding in C and C++ presents

hundreds of examples of secure code, insecure code, and exploits, implemented for Windows and Linux.

So what are the characteristics of C that make it prone to security flaws and what should developers be aware of? Seacord begins to address these questions with specific code examples in the following material that is adapted from his book. More information about Secure Coding in C and C++ is available on the SEI website.

—Richard Lynch, news@sei editor

The C language was created in the early 1970s as a system implementation language for the UNIX operating system. C was derived from the typeless language B [Johnson 73], which in turn was derived from BCPL [Richards 79]. BCPL was designed by Martin Richards in the 1960s and used during the early 1970s on several projects. B can be thought of as C without types or, more accurately, BCPL refined and compressed into 8K bytes of memory.

One goal of a high-level programming language is to provide portability. Portability was not a major goal at the inception of the C programming language but gradually became important as the language was ported to different platforms and eventually became standardized. Portability requires that logic be encoded at a level of abstraction independent of the underlying machine architecture and transformed or compiled into the underlying representation. Problems arise from an imprecise understanding of the semantics of these logical abstractions and how they translate into machine-level instructions. This lack of understanding leads to mismatched assumptions, security flaws, and vulnerabilities.

The C programming language is intended to be a lightweight language with a small footprint. This characteristic of C leads to vulnerabilities when programmers fail to implement required logic because they assume it is handled by C (but it is not). This problem is magnified when programmers are already familiar with superficially similar languages such as Java, Pascal, or Ada, leading them to believe that C protects the programmer better than it actually does. These false assumptions have led to programmers failing to prevent writing beyond the boundaries of an array, failing to catch integer overflows and truncations, and calling functions with the wrong number of arguments.

Another characteristic of C worth mentioning is the lack of type safety. In general, type safety implies that any operation on a particular type results in another value of that type. C was derived from two typeless languages and still shows many characteristics of a typeless or weakly typed language. For example, it is possible to use an explicit cast in C to convert from a pointer to one type to a pointer to a different type. If the resulting pointer is de-referenced, the results are undefined. Operations can legally act on signed and unsigned integers of differing lengths using implicit conversions and producing unrepresentable results. This lack of type safety leads to a wide range of security flaws and vulnerabilities.

An example of an unsafe type in C and C++ programming is C-style strings. Strings—such as command-line arguments, environment variables, and console input—are of special concern in secure programming because they comprise most of the data exchanged between an end user and a software system. Graphic and Web-based applications make extensive use of text input fields and, because of standards like XML, data exchanged between programs is increasingly in string form as well. As a result, weaknesses in string representation, string management, and string manipulation have led to a broad range of software vulnerabilities and exploits.

Strings are a fundamental concept in software engineering, but they are not a built-in type in C or C++. C-style strings consist of a contiguous sequence of characters terminated by and including the first null character. A pointer to a string points to its initial character. The length of a string is the number of bytes preceding the null character, and the value of a string is the sequence of the values of the contained characters, in order.

Common String Manipulation Errors

Programming with C-style strings in C or C++ is error prone. The most common

errors leading to software vulnerabilities are unbounded string copies, null-termination errors, and string truncation.

Unbounded String Copies

Unbounded string copies occur when data is copied from an unbounded source to a fixed length character array (for example, when reading from standard input into a fixed length buffer). In the following C program listing, the program reads characters from standard input using the `gets()` function (on line 4) into a fixed length character array until a newline character is read or an end-of-file (EOF) condition is encountered.

```
1. void main(void) {
2. char Password[80];
3. puts("Enter 8 character password:");
4. gets(Password);
...
5. }
```

Reading data from unbounded sources creates an interesting problem for a programmer. Because it is not possible to know beforehand how many characters a user will supply, it is not possible to pre-allocate an array of sufficient length. A common solution is to statically allocate an array that is much larger than needed. In this example, the programmer is only expecting the user to enter 8 characters, so it is reasonable to assume that the 80-character length will not be exceeded. With friendly users, this approach works well. But with malicious users, a fixed-length character array can be easily exceeded.

It is also easy to make errors when copying and concatenating strings because the standard `strcpy()` and `strcat()` functions perform unbounded copy operations. In the following code listing, the command-line argument in `argv[1]` is copied into the fixed-length static array `name` (line 3).

```
1. int main(int argc, char *argv[]) {
2. char name[2048];
3. strcpy(name, argv[1]);
4. strcat(name, " = ");
5. strcat(name, argv[2]);
...
6. }
```

The static string " = " is concatenated after `argv[1]` in `name` (line 4). A second command-line argument (`argv[2]`) is concatenated after the static text (line 5). Can you tell which of these string copy and concatenation operations may write outside the bounds of the statically allocated character array? The answer, of course, is all of them.

A simple solution is to test the length of the input using `strlen()` and dynamically allocate the memory, as shown in the following code listing:

```
1. int main(int argc, char *argv[]) {
2. char *buff = (char *)malloc(strlen(argv[1])+1);
3. if (buff != NULL) {
4. strcpy(buff, argv[1]);
5. printf("argv[1] = %s.\n", buff);
6. }
7. else {
/* Couldn't get the memory - recover */
8. }
9. return 0;
10. }
```

The call to `malloc()` on line 2 ensures that sufficient space is allocated to hold the command line argument `argv[1]` and a trailing null byte. The `strdup()` function can also be used on Single UNIX Specification, Version 2 compliant systems. The

`strdup()` function accepts a pointer to a string and returns a pointer a duplicate string. The `strdup()` function allocates memory for the duplicate string. This memory can be reclaimed by passing the return pointer to `free()`.

Unbounded string copies are not limited to the C programming language. For example, if a user inputs more than 11 characters into the C++ program shown the following C++ code listing, it will result in an out-of-bounds write.

```
1. #include <iostream.h>
2. int main() {
3. char buf[12];
4. cin >> buf;
5. cout << "echo: " << buf << endl;
6. }
```

Null-Termination Errors

Another common problem with C-style strings is a failure to properly null terminate. In the following code listing, the character arrays `a[]`, `b[]`, and `c[]`are declared as fixed length character arrays.

```
1.int main(int argc, char* argv[]) {
2. char a[16];
3. char b[16];
4. char c[32];
5. strncpy(a, "0123456789abcdef", sizeof(a));
6. strncpy(b, "0123456789abcdef", sizeof(b));
7. strncpy(c, a, sizeof(c));
8. }
```

According to the C99 C language standard (ISO/IEC 9899:1999), the `strncpy()` function copies no more than n characters from a source array to a destination array. Therefore, if there is no null character in the first n characters of the destination array, the resulting string is not null-terminated. As a result, neither `a[]` nor `b[]` is properly terminated in the above example. Null-termination errors are difficult to detect and can lie dormant in deployed code until a particular set of inputs causes a failure.

String Truncation

String truncation occurs when a destination character array is not large enough to hold the contents of a string. It may occur while reading user input or copying a string or even by limiting the number of characters to prevent a buffer overflow. While generally preferable to a buffer overflow, string truncation results in a loss of data, and in some cases can lead to software vulnerabilities.

Mitigation Strategies

Unbounded string copies, null-termination errors, and string-truncation errors have led to numerous vulnerabilities in C and C++ programs. There are, however, numerous mitigation strategies that can be employed to produce more secure code.

C++ programmers, for example, have the option of using the standard `std::string` class. The `std::string` class is the char instantiation of the `std::basic_string` template class, and it uses a dynamic approach to strings in that memory is allocated as required—meaning that in all cases, `size() <= capacity()`. The `std::string` class is convenient because the language supports the class directly. Also, many existing libraries already use this class, which simplifies integration. Problems can still arise in converting from `basic_string` to C-style strings and in using the subscript operator `[]` (which does not perform bounds checking). However, `basic_string` is generally less prone to errors that result in security vulnerabilities.

For C users, the solution is not as obvious. Conventional solutions such as the use of `strncpy()` and `strncat()` are still prone to buffer overflows and truncation errors. As a result, Microsoft has deprecated the use of these functions in Visual Studio 2005. Microsoft developed a set of string functions that can be effective in the remediation of legacy code and submitted them to the ISO/IEC SC22 WG14 J11 International

Standardization Working Group for the Programming Language C. The CERT/CC is also working on a managed string library for C that provides capabilities similar to those of the `basic_string` class in C++. [Long 05, Seacord 05]. An alpha version of this library will be available from www.cert.org in the first quarter of 2006 for testing.

Summary

C is a popular and viable language although it has characteristics that make it prone to security flaws. Some of these problems may be addressed as the language standard, compilers, and tools evolve. In the short term, the best hope for improvement is to educate developers in how to program securely by recognizing common security flaws and applying appropriate mitigations. In the long term, improvements must be made in the C language standard and implemented in compliant compilers for C to remain a viable language for developing secure systems.

References

[Richards 79]
Richards, Martin and Whitby-Strevens, Colin. BCPL: The Language and Its Compiler. New York, NY: Cambridge University Press, 1979 (0-521-21965-5).

[Johnson 73]
Johnson, S. C. and Kernighan, B.W. The Programming Language B (Computing Science Technical Report No. 8). Murray Hill, NJ: Bell Labs, 1973.

[Long 05]
Fred Long, Robert C. Seacord. Specification for Managed Strings.

[Seacord 05]
Robert Seacord. "Managed String Library for C." *C/C++ Users Journal 23*, 10 (October 2005): 30-34.

About the Author

Robert Seacord began programming professionally for IBM in 1982 and has been programming in C since 1985, and in C++ since 1992. He is currently a senior vulnerability analyst with the CERT/Coordination Center (CERT/CC) at the Software Engineering Institute (SEI). As a member of the Vulnerability Analysis Team, Seacord works with other CERT team members to analyze software vulnerability reports and assess the risk to the Internet and other critical infrastructures, identify underlying causes of vulnerabilities, and develop coding practices to improve the security of software systems. He is coauthor of two other books: *Building Systems from Commercial Components* (Addison-Wesley, 2002) and *Modernizing Legacy Systems* (Addison-Wesley, 2003).

Integrating Architecture Methods: The Case of the QAW and the ADD Method

NEWS AT SEI

Authors

Paul C. Clements

Robert Nord

This library item is related to the following area(s) of work:

Software Architecture

This article was originally published in News at SEI on: March 1, 2005

In a previous column ("Rethinking the Software Life Cycle"), we looked at the traditional software-development life cycle in the context of the architecture-centric methods that we have developed at the Carnegie Mellon Software Engineering Institute (SEI) over the past 10 years. These methods include the SEI Architecture Tradeoff Analysis Method (ATAM) [Clements 02], the SEI Quality Attribute Workshop (QAW) [Barbacci 03], the SEI Attribute-Driven Design (ADD) Method [Bass 03], the SEI Cost Benefit Analysis Method (CBAM) [Bass 03], and SEI Active Reviews for Intermediate Design (ARID) [Clements 02].

This column concentrates on the QAW and the ADD methods to show how they can be enhanced and integrated with a software-development life-cycle process to help organizations methodically design complex software-intensive systems. The QAW provides a way to elicit and articulate detailed quality-attribute requirements for a system, which the architecture must support. ADD is an architectural design method that starts with statements of quality-attribute requirements and guides the architect through a series of design decisions that help to meet those requirements. As such, these two methods can be integrated.

There are three ways that the QAW and the ADD method can be made to work together:

1. tailor QAW knowing that it will be followed by the ADD method

Related Links

News

SATURN Conference Announces Additional Keynote, Conference Scholarships

Distinguished Speakers, Strong Technical Program Set for SATURN 2014

See more related news »

Training

Big Data - Architectures and Technologies

Documenting Software Architectures - eLearning

See more related courses »

2. tailor ADD making use of the knowledge gained by conducting a QAW
3. add activities to bridge the QAW and the ADD methods

Knowing that ADD is to follow, we can tailor the QAW in the following ways:

- The information needed by the ADD method could constrain the QAW in terms of the types of scenarios generated, the stakeholders invited, the questions or issues recorded in the refinement template, and so forth.
- The notion of general scenarios was originally developed in the context of ADD. Incorporating them into the QAW would help guide scenario refinement into six parts.
- The last step of the ADD method refines the quality-attribute scenarios in preparation for the next iteration of the method on the modules of the decomposed system. Such activities might be applied to refine the system scenarios that flow from the QAW to the ADD method.

Knowing that a QAW has already been conducted, we can tailor the ADD method in the following ways:

- An enhanced QAW could help guide or take the place of the ADD step in which the architectural drivers are chosen. Mapping quality attributes to business goals is necessary to support the construction of a utility tree before applying the ADD method.
- High-priority scenarios from the QAW could inform the types of views that are selected, which would encourage designers to look at the difficult design problems first.
- In addition to quality requirements, the ADD method also takes constraints and functional requirements as inputs. QAW scenarios could help identify use cases, which are input into ADD as functional requirements. Issues and questions in refined scenarios that are further refined into themes and suggestions could provide input into ADD as constraints.

Additional activities to bridge the methods would follow the QAW in preparation for applying the ADD method:

- Analyze the QAW results.
- Hold a post-QAW planning workshop.
- Transform the elicited scenarios to be useful for ADD.

The results should be analyzed immediately after the QAW, and the post-QAW planning workshop should follow the analysis. The transformation of the scenarios could occur during the action-planning workshop. Other requirements analysis would occur in parallel with the QAW, for example, refinement of the functional requirements into the form of use cases. Scenarios generated during the QAW could lead to multiple use cases.

At the completion of the QAW, information associated with the quality attributes will have been generated and prioritized, but this information will not have been put into context. The first step toward providing this context would be to analyze the QAW results. The suggested steps for this analysis are described below:

1. Build an initial utility tree showing how the scenarios are organized according to the quality attributes. The scenarios of the utility tree contain an implicit measure of “business importance” from the stakeholder voting. This needs to be cross checked with the business manager. Note that the “architectural difficulty” of the scenarios would be assigned by the architects at a later time (e.g., during the post-QAW planning workshop or during the application of the ADD method).
2. Build a table showing how the scenarios map to the business goals, and identify any missing business goals that are obvious from the QAW results.
3. Build a table showing how the scenarios map to the architectural drivers, and identify any missing architectural drivers.

4. Build a table showing the architectural life-cycle processes being used and how they relate to the scenarios. Identify any missing processes implied by a scenario.
5. Develop a list of concerns resulting from the missing business goals, architectural drivers, and architectural life-cycle processes. Note concerns resulting from the lack of scenario coverage in any of the business goals, architectural drivers, and processes identified earlier.
6. Identify potential studies required and prototype development necessary to mitigate the risks and concerns that have been identified.

After the QAW results have been analyzed, a post-QAW planning workshop can be held to review concerns expressed in the results analysis and determine what further actions can be taken to aid the ADD approach. Suggested activities for such a workshop are listed below:

1. Review the concerns and classify them in some manner: for example, those that can be dealt with in a cursory manner (e.g., by adding a new business goal), those that require some further exploration (e.g., by writing appropriate white papers or doing feasibility studies), and those that require extensive integration with outside agencies.
2. Determine if any of the lower priority scenarios should be refined, and refine the appropriate scenarios.
3. Have the architects add a difficulty ranking to the scenarios in the utility tree.
4. Perform flow downs of the scenarios.

Some scenarios from QAW may be useful directly in ADD. Other scenarios may need to be refined and allocated to hardware, software, people, or data in the system architecture. The ADD method can be used on the software-architecture portion of the system architecture.

The QAW stakeholders often express high-level scenarios of the following types:

- operational scenarios. These scenarios describe, for example, how a system of systems responds to a stimulus while under a heavy workload.
- life-cycle process scenarios. These scenarios are largely concerned with the life-cycle processes involved with the system of systems.

A QAW scenario may not distinguish between what is done by operators and what is done by the automation, especially where multiple operators are involved in the scenario. In this case, the architect will have to review the existing documentation (such as those documents listed above) and transform the original scenario into a group of overlapping scenarios, which combine to provide the original scenario. Some examples of such transformations are as follows:

- A system-specific quality scenario describes the operation between a sensor detecting an event of interest and an actuator responding to the event, including a time deadline for the response; however, it ignores the manual operations involved. The architect reviews the other documentation and discovers that the scenario involves a number of operators at a number of locations using different systems. The architect must then re-express this scenario as a number of scenarios that can be used for the development of the software architecture for the interoperation between the systems, allocating timing budgets to each scenario to satisfy the original timing deadline.
- A system-specific quality scenario describes an activity involving many (hundreds of) nodes connected by a low-bandwidth communication system. The software architect knows enough from the scenario to separate the software architectural decisions from the system decisions and to build a software-architectural fragment to satisfy the scenario. However, the architect cannot verify that the scenario will be satisfied until a simulation study is executed for the environmental workload included in the scenario. In this case, he or she will probably have to work cooperatively with the system engineers.
- A 30-day quick-update requirement is levied on the system to correct some

inadvertent behavior. The software architect must then describe how this requirement affects the architecture, development, integration, test, and field-deployment organizations.

The ADD method has an activity for refining use cases, constraints, and scenarios that could be applicable here. Requirements and constraints flow from the module chosen to decompose to its child modules. Quality scenarios also have to be refined and assigned to the child modules. This is done in one of the following ways:

- A quality scenario may be completely satisfied by the decomposition without any additional impact. It can then be marked as satisfied.
- A quality scenario may be satisfied by the current decomposition with constraints on child modules.
- The decomposition may be neutral with respect to a quality scenario. This scenario should be assigned to one of the child modules.
- A quality scenario may not be satisfied with the current decomposition. If the scenario is important, the decomposition should be reconsidered. Otherwise, the rationale for the fact that the decomposition does not support this scenario must be recorded. This occurrence is usually the result of a tradeoff with other, perhaps higher priority, scenarios.

More details of this approach can be found in *Integrating the Quality Attribute Workshop (QAW) and the Attribute-Driven Design (ADD) Method* (CMU/SEI-2004-TN-017) by Nord et al. [Nord 04].

The benefits of such an integrated method would be the combination of otherwise duplicative steps, more productive use of stakeholders' scarce time, more timely collection of necessary information, and more effective use of that information to achieve the desired architectural outcomes. Some work has already been done to demonstrate the benefits of such an approach. General scenarios are routinely used to help guide the refinement of scenarios during the QAW. A utility tree is constructed after the QAW if requested by the sponsor or if a follow-on ATAM is contemplated. The integrated approach has been applied to the design of patient-monitor management software [Lounento 05]. Further work remains to be done to verify the benefits of our proposed integration approach through pilot projects with customers. The goal of such projects is to provide tailored architecture methods to help customers add architecture-based and quality-attribute-based thinking to their planning and development efforts.

References

[Barbacci 03]
Barbacci, M. R.; Ellison, R.; Lattanze, A. J.; Stafford, J. A.; Weinstock, C. B.; & Wood, W. G. Quality Attribute Workshops (QAWs), Third Edition (CMU/SEI-2003-TR-016). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003.

[Bass 03]
Bass, L.; Clements, P.; & Kazman, R. *Software Architecture in Practice, 2nd edition*. Boston, MA: Addison-Wesley, 2003.

[Clements 02]
Clements, P.; Kazman, R.; & Klein, M. *Evaluating Software Architectures: Methods and Case Studies*. Boston, MA: Addison-Wesley, 2002.

[Lounento 05]
Lounento, H. *Quality Attribute Based Architecture Design of Patient Monitor Management Software* (Master's Thesis). Helsinki, Finland: Laboratory of Software Business and Engineering, Department of Computer Science and Engineering, Helsinki University of Technology, 2005.

[Nord 04]
Nord, R. L.; Wood, W. G.; & Clements, P. C. Integrating the Quality Attribute Workshop (QAW) and the Attribute-Driven Design (ADD) Method (CMU/SEI-2004-TN-017). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University,

2004.

About the Authors

Robert Nord is a senior member of the technical staff in the Product Line Systems Program at the SEI where he works to develop and communicate effective methods and practices for software architecture. Prior to joining the SEI, he was a member of the software architecture program at Siemens, where he balanced research in software architecture with work in designing and evaluating large-scale systems. He earned a PhD in computer science from Carnegie Mellon University. Nord lectures on architecture-centric approaches. He is co-author of *Applied Software Architecture* (1999) and *Documenting Software Architectures: Views and Beyond* (2002).

William Wood has been a member of technical staff at the SEI for 18 years. During this time he has managed a technical program and technical projects and provided technical support to the program development organization. He is currently working in software architecture with a number of clients. Previously Wood worked in process control automation for Westinghouse Electric Corp. for 20 years. He has an MSEE degree from Carnegie Mellon University, and a BSc in physics from Glasgow University, Scotland.

Paul Clements is a senior member of the technical staff at the SEI, where he has worked for 10 years leading or co-leading projects in software product line engineering and software architecture design, documentation, and analysis. Clements is the co-author of three practitioner-oriented books about software architecture: *Software Architecture in Practice* (1998; second edition, 2003), *Evaluating Software Architectures: Methods and Case Studies* (2001), and *Documenting Software Architectures: View and Beyond* (2002). He also co-wrote *Software Product Lines: Practices and Patterns* (2001), and was co-author and editor of *Constructing Superior Software* (1999). In addition, Clements has also written dozens of papers in software engineering reflecting his long-standing interest in the design and specification of challenging software systems. He received a BS in mathematical sciences in 1977 and an MS in computer science in 1980, both from the University of North Carolina at Chapel Hill. He received a PhD in computer sciences from the University of Texas at Austin in 1994.

The views expressed in this article are the author's only and do not represent directly or imply any official position or view of the Software Engineering Institute or Carnegie Mellon University. This article is intended to stimulate further discussion about this topic.

Analyzing the Reuse Potential of Migrating Legacy Components to a Service-Oriented Architecture

NEWS AT SEI

Authors

Grace Lewis

Edwin J. Morris

Dennis B. Smith

This library item is related to the following area(s) of work:

Performance and Dependability

This article was originally published in News at SEI on: March 1, 2005

In the previous column, we discussed the important role of service-oriented architectures, especially as an enabler for interoperability. With the advent of universal Internet availability, many organizations have leveraged the value of their legacy systems by exposing all or parts of their functionality as services. A service is a coarse-grained, discoverable, and self-contained software entity that interacts with applications and other services through a loosely coupled, often asynchronous, message-based communication model [Brown 02]. A collection of services with well-defined interfaces and a shared communications model is called a service-oriented architecture (SOA). A system or application is designed and implemented as a set of interactions among these services.

The characteristics of SOAs (e.g., loose coupling, published interfaces, and standard communication model) offer the promise of enabling existing legacy systems to expose their functionality, presumably without making significant changes to the legacy systems [Lewis 05]. SOA migration tasks can be considered from a number of perspectives including that of the end client or user of the services, the SOA architect, or the service provider. This column focuses on the service provider.

Related Links

Training

Software Architecture Design and Analysis

Modeling System Architectures Using the Architecture Analysis and Design Language (AADL)

See more related courses »

Organizations often look toward developing services from legacy components. However, constructing services from existing systems to obtain the benefits of an SOA is neither easy nor automatic. In fact, such a migration can represent a complex engineering task, particularly when the services are expected to execute within a tightly constrained environment.

One important task is to evaluate legacy components to determine their potential for migration to services within a specific SOA. This column outlines an approach, the Service-Oriented Migration and Reuse Technique (SMART), that gathers information and identifies the risks for such a migration effort in a systematic way.

Creation of Services from Legacy Components

Enabling a legacy system to interact within a service-oriented architecture, such as a Web-services architecture, is sometimes relatively straightforward—this is a primary attraction to the approach for many businesses. However, characteristics of legacy systems such as age, language, and architecture, as well as of the target SOA, can complicate the task. An analysis must be performed to consider

1. requirements from potential service users (It is important to know what applications would use the services and how they would be used. For example, what is the information expected to be exchanged? In what format?)
2. technical characteristics of the target environment, such as bindings, messaging technologies, communication protocols, service-description languages, and service-discovery mechanisms
3. the architecture of the legacy system, including dependencies on commercial products or specific operating systems, or poor separation of concerns
4. the effort involved in writing the service interface
5. the effort involved in the translation of data types
6. the effort required to describe the services including information about qualities of service, such as performance, reliability, and security; or service-level agreements (SLAs)
7. the effort involved in writing service-initialization code and operational procedures
8. estimates of cost, difficulty, and risk

SMART performs these types of analyses through five activities:

1. Establish stakeholder context.
2. Describe existing capabilities.
3. Describe the future service-based state.
4. Analyze the gap between service-based state and existing capabilities.
5. Develop strategy for service migration.

These five activities are briefly outlined below.

1. **Establish Stakeholder Context**
   To establish the context in which the migration to services will take place, SMART first identifies the stakeholders, including the current end users of the legacy systems, the potential end users of the migrated service operating within the SOA, and the owners of the legacy systems. The activity identifies who knows most about the legacy system, what it currently does, and what it should do as a service or set of services.
2. **Describe Existing Capabilities**
   The goal of the second activity is to obtain descriptive data about the legacy components. Basic data solicited includes the name, function, size, language, operating platform, and age of the legacy components. Technical personnel provide information about the architecture, design paradigms, code complexity, level of documentation, module coupling, interfaces for systems and users, and dependencies on other components and commercial products. Historical cost

data for development and maintenance tasks is collected to support effort and cost estimates.

3. **Describe the Future Service-Based State**
   The goals of the third activity are to

   - gather evidence about potential services that can be created from the legacy components
   - gather sufficient detail about the target SOA to support decisions about what services may be appropriate and how they will interact with the architecture

   Initial information about potential services often comes from conversations about the function(s) of the legacy system during the second activity. However, the information gathered often must be tempered by data from users, corporate architects, domain groups, communities of interest, and reference models that address service definition. In some cases, these groups and models will define the entire set of services that support the organization's goals and into which any potential services built from the legacy components must fit.

4. **Analyze the Gap**
   The goal of the fourth activity is to identify the gap between the existing state and the future state and determine the level of effort needed to convert the legacy components into services. This analysis may also suggest potential tradeoffs between the target architecture and the legacy components.

   The tasks of this activity include the following:

   - Develop an analysis strategy for legacy components that are being considered for migration.
   - Analyze the legacy components to determine the types of changes that must be made to enable migration. SMART uses three sources of information to support the analysis activity:

     1. the issues, problems, and other concerns that were noted as the team completed the previous, discovery-oriented steps form one source of information.
     2. a Service Migration Inventory (SMI) that distills the many desired traits of services executing within SOAs into a set of topics. The team uses the SMI to assure broad coverage and consistent analysis of difficulty, risk, and cost issues.
     3. (optional) the use of code analysis and architecture reconstruction tools to analyze the existing source code for legacy components.

5. **Develop Strategy for Service Migration**
   A key feature of SMART involves building cost projections for each migration option still under consideration. This is accomplished by considering organizational characteristics, difficulty, and risk associated with various migration options and applying historical productivity numbers where possible.

Pilot Application of the Process

We applied an early version of SMART in a recent pilot analysis of the potential for migrating a set of legacy components from a DoD command and control (C2) system to an SOA.

1. Establish Stakeholder Context

We initially met with the government owners of the system and the contractors who had developed the system. At this meeting we received an overview of the set of systems, the history of the systems, the migration plans, and the drivers for the migration. We received a brief orientation to the SOA and also obtained system documentation.

The owners of the systems recognized that if selected components from their C2 system were to be converted to application domain services within a specific target SOA, they could have applicability for a broad variety of purposes. Our role was to perform a preliminary evaluation of the feasibility of converting a set of their components to application-domain services within an SOA.

2. Describe Existing Capabilities

The pilot C2 system has two parts: (1) a mission-planning system and (2) a mission-execution system that adds situational awareness to the planning capability. These two systems were initially developed as part of a product line. Both rely on a set of core components for the data model, data analysis, and visualization.

Given the information about the target SOA, we met with the contractor and representatives of the government to focus on a limited number of legacy components and to select criteria for further screening. We focused on seven potential services that the government team had previously identified as part of its initial analysis of ADS requirements. These seven potential services contained 29 classes.

The current system, written in C++ on a Windows operating system, had a total of about 800,000 lines of code and 2500 classes. In addition, the system had dependencies on a commercial database and a second product for visualizing, creating, and managing maps. Both commercial products have only Windows versions.

The 29 classes that we selected enabled us to focus on potentials for high payoff. In conjunction with the team, we developed criteria for screening the potential reusable components. These criteria included

- size
- complexity
- level of documentation
- coupling
- cohesion
- number of base classes
- programming-standards compliance
- black-box vs. white-box suitability
- scale of changes required
- commercial-mapping-software dependency
- Microsoft dependency
- support software required

These criteria formed the basis for the more detailed analysis discussed below.

3. Describe the Future Service-Based State

The system owner had done a preliminary identification of potential services that could be built from components of the legacy system. This analysis was derived from high-level requirements for applications that were being targeted as users of services to be provided by the SOA. The system owner had matched legacy functionality to these high-level requirements and provided some initial estimates of the contents of the potential services.

We investigated the target SOA through an analysis of available documentation and through a meeting with the developers. Because the SOA was still under development, the specifications for how to deploy and write services were still unclear.

4. Analyze the Gap

Given the known and projected constraints of the target SOA, we performed three different types of analyses: (1) an analysis of the changes to the legacy components that would be necessary for migration to the SOA, (2) an informal evaluation of code quality, and (3) an architecture reconstruction to obtain a better understanding of the set of undocumented dependencies. The results of these analyses allowed us to define a service-migration strategy based on the risks resulting from the unknown future state of the target SOA.

Analysis of Required Changes
We initially met with the contractor to get an understanding of the required changes, as well as estimates of the level of difficulty and the risks of making the changes. The contractor provided estimates for converting the components into services, based on a set of simplifying assumptions on the actual make-up of the target SOA and the final set of user requirements. These estimates initially suggested that the level of difficulty of making these changes would be low to medium, and the risk would be low because of the contractor's familiarity with the systems.

However, we found that the tools in use on the project picked up only first-level dependencies between classes. This indicated that the coupling and the amount of code that was used by each class was higher than could be estimated from the existing documentation. There was also no consistent programming standard, leading to idiosyncrasies among programmers. Because of the inadequacies that we found in the architecture documentation and the underestimation of the amount of code used by the potential services, there remained a number of gaps in our understanding of the system.

Code Analysis
To address remaining issues, we first analyzed the code through a code analyzer "Understand for C++."

The code analysis enabled us to validate the input from the contractor and to produce input for the architecture reconstruction tool that would identify dependencies.

From the code analysis, we found that the code was better organized and documented at the code level than most code that we have seen. However, there were inconsistencies in the quality and documentation among different parts of the code that made the analysis complicated.

Architecture Reconstruction
To address the issue of dependencies in more detail, we conducted an architecture reconstruction with a tool called ARMIN. Architecture reconstruction is the process by which the architecture of an implemented system is obtained from the existing system [Kazman 03].

In our analysis, we were interested in dependencies between

- services and user interface classes
- services and the commercial mapping software
- services
- the services and the rest of the code that mainly represented the data model

The architecture reconstruction identified a substantial number of undocumented dependencies between classes. These will enable a more realistic understanding of the scope of the migration effort if it succeeds.

The architecture reconstruction also enabled us to document the central role of the data model and to identify it as a potentially valuable reusable component even though it had not been identified during the initial analysis.

5. Develop Strategy for Service Migration
In looking at the potential for reuse of the existing legacy components, we found that the current legacy code represents a set of components with significant reuse potential. However, because the current legacy system does not have sufficient architecture or other high-level documentation, it was difficult to understand the big picture as well as

dependencies between different classes. The largest risk in reusing the legacy components concerns the fact that the SOA has not been fully developed. We also recommended that the government organization require the following changes from its contractors to make reuse of its legacy components more viable:

- a suitable set of architectural views
- consistent use of programming standards
- documentation of code to enable comments to be extracted using an automated tool
- documentation of dependencies, especially when they violate architecture paradigms

Conclusions and Next Steps

We found that the initial task of determining how to expose functionality as services, while seemingly straightforward, can have substantial complexity. Our conclusions, while not definitive, did point out a number of issues that the client had not previously considered. The type of disciplined analysis that we performed appears to have applicability for other organizations considering migrations to SOAs.

References

[Brown 02]
Brown, A; Johnston, S.; and Kelly, K. *Using Service-Oriented Architecture and Component-Based Development to Build Web Service Applications*. Rational Software Corporation, 2002.

[Kazman 03]
Kazman, R; O'Brien, L.; and Verhoef, C. *Architecture Reconstruction Guidelines, 2nd Edition* (CMU/SEI-2002-TR-034). Software Engineering Institute, Carnegie Mellon University, 2003.

[Lewis 05]
Lewis, Grace and Wrage, Lutz. *Approaches to Constructive Interoperability* (CMU/SEI-2004-TR-020). Software Engineering Institute, Carnegie Mellon University, 2005.

About the Authors

Grace Lewis is a senior member of technical staff at the Software Engineering Institute (SEI) of Carnegie Mellon University (CMU), where she is currently working in the areas of constructive interoperability, commercial-off-the-shelf- (COTS-) based systems, modernization of legacy systems, enterprise information systems, and model-driven architecture. Her latest publications include several reports published by Carnegie Mellon on these subjects and a book in the SEI Software Engineering Series. Grace has more than 15 years of experience in software engineering. She is also a member of the technical faculty for the Master in Software Engineering program at CMU.

Edwin Morris is a senior member of the technical staff at the Software Engineering Institute, assigned to the Integration of Software-Intensive Systems (ISIS) Initiative. He is currently investigating approaches to achieving technical interoperability between complex systems and programmatic interoperability between the organizations that build and maintain them. Previous activities involved improving processes and techniques for the evaluation and selection of COTS products, and the development of the COTS Usage Risk Evaluation (CURE) technology. Before coming to the SEI, Morris developed custom operating systems for embedded microprocessors along with support tools to predict and monitor the performance of real time systems.

Dennis Smith is the lead for the SEI Integration of Software Intensive Systems Initiative. This initiative focuses on addressing issues of interoperability and integration in large-scale systems and systems of systems. Earlier, he was the technical lead in the effort for migrating legacy systems to product lines. In this role he developed the method Options Analysis for Reengineering (OARS) to support reuse decision-making. Smith has also been the project leader for the CASE environments project. This project examined the underlying issues of CASE integration, process

support for environments and the adoption of technology. Smith has published a wide variety of articles and technical reports, and has given talks and keynotes at a number of conferences and workshops. He has an MA and PhD from Princeton University, and a BA from Columbia University.

The views expressed in this article are the author's only and do not represent directly or imply any official position or view of the Software Engineering Institute or Carnegie Mellon University. This article is intended to stimulate further discussion about this topic.

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

Large-Scale Work—Part II: The Project

NEWS AT SEI

Author

Watts S. Humphrey

This library item is related to the following area(s) of work:

Process Improvement

This article was originally published in News at SEI on: March 1, 2005

This is the second column in a series on large-scale development work. The first column briefly discussed several important large-scale-project issues and then addressed the problems of organization size. It described how organizations get more complex as they grow and how this complexity saps the energy that the organization has available for productive work. I discussed the reasons for these problems as well as their consequences for development projects. In this column, I continue this discussion with primary focus on the project itself and the principal project-management issues. Large projects generally exist in large organizations, and they are also usually spread across multiple departments and locations. This affects their ability to operate and to produce high-quality products for predictable costs and schedules.

Large-Systems Problems

While large-scale projects face a host of problems, the principal topics I will discuss are project management and people management. Technical issues are critically important, and no amount of process, project, and people management can compensate for poor technical work, but this subject is too vast to usefully discuss in a brief column. Therefore, this column focuses on project management, and subsequent columns will deal with the people-management topics in managing large, distributed, multi-organizational projects. The major problems I have seen concern delegation and coordination.

A War Story

One of the best examples I know of the typical problems of managing large projects is a situation that developed shortly after I took over as IBM's director of programming.

Related Links

News

SEI to Co-Sponsor 26th Annual IEEE Software Technology Conference

SEI Cosponsors Agile for Government Summit

Training

See more related courses »

Events

Team Software Process (TSP) Symposium 2014

Nov 3 - 6

Our two largest locations were in Poughkeepsie, N.Y., and San Jose, Calif. Because the Poughkeepsie group had grown to nearly 1,000 developers and because it still needed many more people to handle its committed work, we had decided to move the data-management mission from Poughkeepsie to San Jose. The problem was that the interface between the Poughkeepsie operating-system work and the San Jose data-management products had not been defined.

We had told the two groups to agree on the interface definition and to come to me if they could not. Unfortunately, the interface definition had become a highly contentious issue with no clearly best answer, and the two groups were unable to agree. One morning, the two laboratory managers with about 30 of their best technical people showed up in my office with two competing proposals. Although I knew less about this subject than anyone else in the room, I ended up designing the interface. I told the groups to either make my definition work or to agree on a better solution and to let me know what they agreed on. They never came back.

This was a management failure. The problem had started many months before when what should have been a technical issue had become highly political. Then the technical experts were essentially forced to defend their managements' positions rather than carefully considering the pros and cons of the various alternate ways to design the interface. We therefore ended up making a political decision on a technical issue.

The Management Error

While the way I handled this interface problem was clearly a management failure, what was the specific failing, and what should I have done differently? In retrospect, I believe that the problem was caused by differing goals and objectives. The Poughkeepsie manager was the OS/360 project manager while the San Jose manager was a laboratory manager with multiple projects, one of which was the OS/360 data-management work. Therefore, they had no common basis for agreeing on what was the best solution. Rather than making an instant decision, I should have made these two groups resolve the problem properly.

I now believe that the proper approach would have been to start with the goals to be met by the decision. That was a topic where I, as the senior manager, could have made a real contribution. Then, after reaching agreement on the goals to be met by the interface decision, we should have defined the criteria for making the decision. After settling the goal and criteria questions, I should then have sent the groups off to make the decision themselves and asked them to either report their conclusion and rationale to me or to come back if they needed further help.

The Generic Problem

The basic problem that this story illustrates is that technically trained managers love to make technical decisions, particularly after they have become executives. We all have this image of decisive executives who can be relied on to make almost instant decisions. This gives them a gratifying sense of power and fosters an image of infallibility. There is this myth that indecision is bad and that projects must have clear and precise direction at all times. This is not only wrong, but dangerous, particularly for very large and highly complex systems.

The design of large systems involves many decisions, most of which could be made in several ways with little, if any, effect on overall system performance. However, a few of these decisions are critical. If these critical decisions are not made properly, the consequences could be severe. Unfortunately, these decisions don't come with a sign that says they are critical. The two examples that come immediately to mind both concern NASA's handling of two space shuttle decisions: the technical question of O-ring safety for a low-temperature launch and the question of potential damage caused when falling hunks of insulation hit the shuttle wing.

The Critical Point

To me, one of the most critical aspects of managing large-scale projects is making sure that decisions are properly made. The executive's responsibility must be to identify the right people to make the decision, insist that the goals used for making the decision be defined and documented, and require that the criteria for the decision be established. While there are far too many technical decisions in large-scale projects for

management to require that they all be made in this way, there are a relatively few times when technical decisions are escalated to senior management. However, whenever they are, these decisions are almost certainly technical issues that have become political.

If the executive does not insist that each of these politically tinged technical decisions is properly made, he or she is likely making a very big and possibly fatal mistake. If ever, this is the one time when the executive should insist that the decision be made in the right way. While these decision situations always come up when there is no time and when everybody, including the executive, is in a rush to get on with the job, this is precisely the time when proper decision making is most important. When executives insist that rush decisions be made in the proper way, they are demonstrating their ability to be technical executives. Therefore, the first two ground rules for the proper management of large-scale projects are the following:

1. Insist that all technical decisions be made by the proper technical people.
2. Make sure that, in making these decisions, the technical decision makers thoughtfully evaluate the available alternatives against clearly defined criteria.

The Broader Lesson

While this might appear to be the end of the story, there is a broader lesson that is even more important. This concerns management development. There are problems at every management level in large organizations. At every level, managers or executives should be most concerned with the way their subordinates operate. The challenge at every level is to delegate and to get those at lower levels to take responsibility for making the decisions that they should make. The managers should insist that their subordinates make all of these decisions and that they make them in the right way.

With few exceptions, the depth of technical knowledge in organizations increases at lower levels of the management hierarchy. This means that when decisions are made with proper goals and sound criteria, they are invariably better decisions when they are made at the lowest possible levels in the organization. So delegation is critical, but with delegation must also come the responsibility to monitor behavior. So, while looking at cost, schedule, and quality performance, executives and managers should also look at the key decisions and how they were made.

It might seem that only senior managers and executives need be concerned with these issues. However, a principal problem for executives in large organizations is the difficulty of getting their technical people to take responsibility and to make sound and timely decisions on their own. The problem, of course, is not just the ability of technical people to take responsibility but also their ability to handle this responsibility responsibly.

In the next column, I discuss how developers and team leaders should act when making technical decisions. I will also outline some principles for team leaders and team members to consider when working on large-scale development projects and some strategies that can help in following these principles.

Acknowledgments

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

In Closing, an Invitation to Readers

In this particular series of columns, I discuss some of the development issues related to large-scale projects. Since this is an enormous subject, I cannot hope to be comprehensive, but I do want to address the issues that you feel strongly about. So, if there are aspects of this subject that you feel are particularly important and would like covered, please drop me a note with your comments, questions, or suggestions. Better yet, include a war story or brief anecdote that illustrates your ideas. I will read your notes and consider them when planning future columns.

Thanks for your attention and please stay tuned in.

Watts S. Humphrey
watts@sei.cmu.edu

About the Author

Watts S. Humphrey founded the Software Process Program at the SEI. He is a fellow of the institute and is a research scientist on its staff. From 1959 to 1986, he was associated with IBM Corporation, where he was director of programming quality and process. His publications include many technical papers and several books. His most recent books are *Introduction to the Team Software Process* (2000) and *Winning With Software: An Executive Strategy* (2002). He holds five U.S. patents. He is a member of the Association for Computing Machinery, a fellow of the Institute for Electrical and Electronics Engineers, and a past member of the Malcolm Baldrige National Quality Award Board of Examiners. He holds a BS in physics from the University of Chicago, an MS in physics from the Illinois Institute of Technology, and an MBA from the University of Chicago.

The views expressed in this article are the author's only and do not represent directly or imply any official position or view of the Software Engineering Institute or Carnegie Mellon University. This article is intended to stimulate further discussion about this topic.

CMMI V1.2: What's Changing? (Part 2)

NEWS AT SEI

Author

Mike Phillips

This library item is related to the following area(s) of work:

CMMI

Process Improvement

This article was originally published in News at SEI on: April 1, 2005

Background-Pre-V1.2 Development

Before Version 1.0, the CMMI Product Team used a large stakeholder group to provide feedback on early draft versions of the model. These early versions were not yet ready for release, and comments from the stakeholder group were often wide-ranging in seeking to set the broad course for combining the best parts of three distinct source models:

- Software CMM Version 2, draft C (this version was never publicly released, but it improved on the widely used SW-CMM Version 1.1)
- EIA Interim (at the time) Standard 731 (this standard brought together two concurrently developed systems engineering models)
- Integrated Product Development CMM (this CMM reached a draft Version 0.98 but was never publicly released)

These were lively days. We wrestled with whether to continue a process area for peer reviews that was strongly desired by the software community and whether to add a process area for data management that was strongly desired by the systems engineering community.

With the baseline release of CMMI Version 1.0, we instituted a more formal mechanism for change. The CMMI Steering Group created a Configuration Control Board (CCB) with a majority of the membership from outside the SEI to help us judge the community's tolerance for-and interest in-change. We formalized the change-

Please note that current and future CMMI research, training, and information has been transitioned to the CMMI Institute, a wholly-owned subsidiary of Carnegie Mellon University.

Related Links

News

SEI to Co-Sponsor 26th Annual IEEE Software Technology Conference

SEI Cosponsors Agile for Government Summit

Training

See more related courses »

Events

Team Software Process (TSP) Symposium 2014

Nov 3 - 6

request system with automated support to help us keep track of the requests for change to CMMI models, the SCAMPI appraisal method, and the CMMI training courses. When we developed the current Version 1.1, we used change requests from all sources and also announced a formal review period to stimulate reviews by various organizations that used such review mechanisms in their standards work.

The criteria that our steering group provided for the development of Version 1.1 meant that many of the aggressive changes recommended by some reviewers would not be allowed for that version. But rather than rejecting those change requests, we kept them in the change-request system and marked them "deferred." These change requests would be reviewed and analyzed again for future product-suite updates.

The CR Process for V1.2

While the release plan for Version 1.1 was to provide a mature, fully operational version that we knew deserved a rather quick release (one year after V1.0), the intent was to then stabilize use of V1.1 for an extended period-three to five years. Our collection of change requests was maintained and began to grow. We allowed for modest change by maintaining a collection of errata. This collection largely consisted of typographical errors, but sometimes included changes to text that we realized was particularly misleading. The publication of CMMI: Guidelines for Process Integration and Product Improvement by Addison-Wesley also allowed the discovery of a few more corrections that we captured in errata and made available on the SEI Web site.

We spent about a year seeking feedback from a wide range of users through the CMMI Interpretive Guidance Project to see which concepts were being easily understood and which were more difficult for adopters. Sometimes the difficult concepts actually involved understanding the differences between concepts in a source model (e.g., Software CMM) and the concepts in the CMMI model; other difficult concepts involved understanding the greater breadth of coverage of the enterprise development effort offered by CMMI.

When we began analyzing the change requests for the model, we had a varied collection of requests. Some were several years old-requests deferred from the V1.1 update; some were notes or issues elicited as part of the CMMI Interpretative Guidance Project; some were received as part of the formal review period; and others were change requests that were submitted by individuals and groups at various times after the release of Version 1.1 as they used the product suite and saw opportunities to improve the material.

The model team examined all of these change requests and addressed as many of them as possible. At the same time, the team also needed to abide by the criteria for Version 1.2 provided by the CMMI Steering Group. These criteria were to assure that the upgrade from existing material does not constitute a radical change in the product suite that might require major revision in existing improvement efforts or additional training for users.

Lessons Learned

One lesson that we've learned from this analysis of change requests is that maintaining deferred change requests can be problematic. Some change requests were deferred from the development of Version 1.02. Could we be sure that someone who expressed the wish for a change in 2002 while using CMMI Version 1.02 would want the same change in 2005 using CMMI Version 1.1 and with more experience and familiarity with the concepts?

Administration of the change-request management system has also had its challenges. We have been successful in assuring that every change package we prepare for CCB consideration is based on submitted change requests, not just a desire of model team members. But often change requests in a given area recommended remedies different from those proposed by the model team in the change package. In some cases, this has meant that a requested change has been approved while the proposed remedy was rejected. Further, it was not uncommon for change requests to have multiple ideas submitted on one form, so that a portion of the change request could be addressed in the current development cycle, while another portion was rejected or deferred. This situation made the dialogues between the model team and the CCB challenging at

times, even though the change-request system allowed for change requests to be subdivided if needed.

To date, we have not been able to determine an efficient method to advise the many change-request submitters about the status of their recommendations. We have, however, been able to send an automated response to each submitter that his or her change request has been received into the system.

A query of our change-request database shows that of 1001 change requests considered for V1.2, 360 were approved to provide inputs to the change packages; 590 were rejected, and 51 were beyond our ability to include in V1.2. The various ways that all, some, or none of the proposed content will be seen in the upcoming version makes describing a specific change request as approved, rejected, or deferred beyond V1.2 difficult to characterize. Often proposed change requests that were "rejected" influenced the wording chosen for "approved" changes.

Summary

We would love to conclude this column with a statement that each of you who devoted time and effort to send us your thoughts for improvement would be able to know precisely where and how your recommendation was accommodated in the update. But as you have read above, we have not created the mechanisms for that level of confirmation. We can assure you that we read, analyzed, considered, and debated all of your submissions and that they influenced the model team in its deliberations. Just as we found many opportunities to improve from V1.1, we know that the future will include further opportunities to enhance the approach to CMMI-based process improvement and enhance the quality of our future and existing software-intensive systems.

In the next column, we'll be able to provide more detail on the changes for V1.2 as we prepare to pilot the proposed update.

References

[EIA 98]
Electronic Industries Alliance. Systems Engineering Capability Model (EIA/IS- 731). Washington, DC, 1998.

[SEI 95]
Software Engineering Institute. *The Capability Maturity Model: Guidelines for Improving the Software Process*. Reading, MA: Addison-Wesley, 1995.

About the Authors

As the director of special projects at the Software Engineering Institute, Mike Phillips leads the Capability Maturity Model Integration (CMMI) project for the SEI. He was previously responsible for transition-enabling activities at the SEI. Prior to his retirement as a colonel from the Air Force, he managed the $36B development program for the B-2 in the B-2 SPO and commanded the 4950th Test Wing at Wright-Patterson AFB, Ohio. In addition to his bachelor's degree in astronautical engineering from the U.S. Air Force Academy, Phillips has master's degrees in nuclear engineering from Georgia Tech, in systems management from the University of Southern California, and in international affairs from Salve Regina College and the Naval War College.

Sandy Shrum is a senior writer/editor at the Software Engineering Institute. Since 1998, she has served on the CMMI Product Team in roles such as author, reviewer, editor, and quality assurance process owner. Sandy also serves on the CMMI configuration control board and is the CMMI communications coordinator. She has over sixteen years experience as a technical communicator in the software industry. Sandy earned her MS in professional writing from Carnegie Mellon in 1988.

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

Governing for Security: Protect Stakeholder Interests

NEWS AT SEI

Author

Julia H. Allen

This article was originally published in News at SEI on: April 1, 2005

Leaders must protect stakeholder interests in a visible and accountable manner as part of responsible enterprise security governance. In this discussion, "enterprise" includes the extended or virtual enterprise comprising all of those entities with electronic access to an organization's networks.

Why is This Important?

According to the Corporate Governance Task Force, "Today's economic environment demands that enterprises in both the public and private sectors reach beyond traditional boundaries. Citizens, customers, educators, suppliers, investors, and other partners are all demanding better custodianship of their information and more access to strategic resources. As enterprises rise to meet this demand, traditional boundaries are disappearing and the premium on information security is rising. Heightened concerns about critical infrastructure protection and national homeland security are accelerating this trend" [CGTF 04].

Failure to protect stakeholder interests with respect to certain categories of information or failure to prevent unauthorized access to personal information may have serious legal consequences. An enterprise-wide approach to security governance can help an organization maintain compliance with new and expanding laws and regulations and avoid legal liability related to statutory or common law.

These laws have all provided regulatory incentives for C-level executives and boards of directors to pay closer attention to the subject of information security and thus information-security—or more broadly enterprise-security—governance. The U.S. Sarbanes-Oxley Act, more than any other current legislation, has had the greatest influence on security governance. This is because the statute makes leaders of public corporations responsible for establishing and maintaining adequate internal controls.

A similar security effect derives from both state and international law. The California Database Protection Act (CA SB 1386; notification of personal security-information breaches) and European Union directives on data protection and privacy and electronic communications are affecting multi-state and multinational organizations [CRS 05].

Stakeholders: Stakeholders may include shareholders, investors, rating agencies, partners, suppliers, vendors, customers, employees, consultants, governments (local, state, national), surrounding communities, citizens, and communities of interest such as certifying bodies and professional associations. As defined by the U.S. General Accounting Office, a stakeholder is "an individual or group with an interest in the success of an organization in delivering intended results and maintaining the viability of the organization's products and services. Stakeholders influence programs, products, and services."[1] Stakeholders have some kind of involvement in the organization which could be an investment, share, concern, right, title or responsibility.

Stakeholder Interests: From an enterprise security perspective, stakeholder interests are likely to include

- accurate reporting on the effectiveness and productivity of the enterprise
- creation, preservation, and enhancement of the organization's reputation
- availability and reliability of services (business resilience)
- demonstrated due diligence with respect to protecting against malicious attacks (internal and external) and accidents that can be anticipated
- ensuring of only authorized access to enterprise information
- protecting the privacy of stakeholder information

Protecting stakeholder interests includes being a responsible citizen when connecting enterprise networks to the global internet. The IIA states "In the modern world, everything business or government does with their information technology becomes part of the global information infrastructure. We must build infrastructure to a very high standard. Attaching weak components to the infrastructure puts your organization as well as your neighbors at risk. Responsible citizens will contribute only sound components to that cooperative infrastructure." [IIA 00]

Stakeholder interests are most effectively protected by selecting a broad set of enterprise security principles, interpreting and tailoring these for the enterprise, and ensuring their use and enforcement in the normal course of business. These actions help to ensure that an organization achieves and sustains a culture of security.

Principle-Based Protection: So what does it mean to protect stakeholder interests with respect to governing for enterprise security? An enterprise must select and institutionalize broad enterprise security principles to fully protect stakeholder interests. *Enacted principles inform decisions thereby influencing strategies, plans, and policies. These actions create a culture of security throughout the enterprise.* We have synthesized several credible and reputable sources of information security principles (see below) to clarify what it means to protect stakeholder interests.

These principles of enterprise security derive from foundational work performed by

- American Chemistry Council [ACC 99, ACC 03]
- Business Software Alliance [BSA 03]
- Corporate Governance Task Force [CGTF 04]
- Corporate Information Security Working Group [CISWG 04a, CISWG 04b]
- Information Systems Security Association [ISSA 04]
- Information Technology Governance Institute [ITGI 01, ITGI 04]
- Institute of Internal Auditors [IIA 01]
- International Standards Organization [ISO 00a, ISO 00b]
- National Association of Corporate Directors [NACD 01]

- National Institute of Standards and Technology [NIST 96, NIST 04]
- Organisation for Economic Co-operation and Development [OECD 02]
- Software Engineering Institute [CMMI 03]

These principles (listed below) represent a composite list; we expect that all principles are not applicable for all organizations. Organizations can use this list to select, interpret, prioritize, deploy, and reinforce statements of enterprise-security principles as manifestations of expected behaviors. To be effective and of greatest value, principle selections should be aligned with business objectives including the requirement to protect all stakeholder interests.

Principle descriptions in most of the references address the security of information. We expand these to address security of the extended enterprise that includes but is broader than information security.

Each of the principles is stated using the present tense, conveying what actions, behaviors, and conditions demonstrate the presence of the principle in the organization's culture and conduct.

Enterprise Security Principles Required to Protect Stakeholder Interests include:

- Accountability
- Adequacy
- Awareness
- Compliance
- Effectiveness
- Ethics
- Inclusion
- Individual
- Equity
- Information
- Sharing
- Measurement
- Perspective/Scope
- Response
- Risk Management

Accountability: The governing body (i.e. board of directors; trustees) is accountable for providing effective oversight of enterprise security. Management is responsible for ensuring effective execution of the agreed-to enterprise security program. Such accountability and responsibility is explicit, defined, acknowledged, and accompanied by the authority to act. Leadership accountability and responsibility for security are visible to all stakeholders.

Leaders (members of governing bodies and managers) possess the necessary knowledge, skills, and abilities to fulfill these responsibilities. Individual roles, responsibilities, authorities, and accountabilities are assigned. Leaders ensure that all stakeholders with access to enterprise networks understand their responsibilities with respect to enterprise security. Chief executives sponsor regular enterprise security evaluations, review the evaluation results with stakeholders as appropriate, and report on performance to the governing body, including a plan for remedial action to address any deficiencies.

Adequacy: "How much security is enough may be one of the most critical—and difficult to answer—questions" [IIA 01]. Investment in enterprise security protection strategies (principles, policies, procedures, processes, controls) is commensurate with risk. Determination of risk is based on the value, sensitivity, and criticality of the asset with respect to its vulnerability to loss, damage, disclosure, or denied/interrupted

access. Probability, frequency, and severity of potential vulnerabilities are considered along with a comparison of the cost to reconstitute the asset versus the cost to protect it. Dan Geer suggests that one indicator for the right amount of security investment is how much collaboration you have, meaning the amount of information you have in play resulting from how open your network is to outside parties [Geer 04b]. Other indicators may include the degree and circumstances of critical asset exposure, or the range of tolerable to intolerable consequences resulting from a realized risk.

Leaders ensure that sufficient resources (people, time, equipment, facilities, funds) are authorized and allocated to achieve and sustain an adequate level of security.

Awareness: Leaders are aware of and understand the need to consider security from an enterprise-wide perspective, thus including it in their governance processes. They understand what actions are necessary to protect shareholder and stakeholder value with respect to security. They understand what enterprise-security actions are necessary to retain current customers and attract new customers.

All stakeholders are aware of enterprise security risks and protection strategies and understand their concomitant roles and responsibilities. Enterprise security awareness is demonstrated by the training and education provided to stakeholders who become authorized users of enterprise networks and by requiring periodic training for continued access. Employee position descriptions and agreements with stakeholders define security roles, responsibilities, skills, certifications, and agreements to comply with policy reflect awareness as well. Performance and partner reviews address how well security responsibilities are fulfilled.

Compliance: Enterprise security protection strategies are in compliance with legal and regulatory requirements, requirements of conducting business, and requirements established by external stakeholders. Oversight and actions necessary to objectively evaluate compliance (such as internal and external audits) are built into the enterprise security program. This includes regular monitoring, review, and reporting of compliance findings to affected and interested parties. Leaders ensure that a plan is developed to address remedial and timely action for any security deficiencies and ensure the plan is effectively executed.

Effectiveness: Actions to achieve and sustain adequate enterprise security are demonstrably aligned with enterprise objectives, critical success factors, and the mitigation of enterprise security risks. Security priorities and resources are determined based on this alignment. As a result, stakeholders view enterprise security in an enabling role, similar to audit, quality assurance, program management, and environmental protection [IIA 01]. Enterprise security is subject to continuous review, periodic testing, and an evaluation of its effectiveness as measured against enterprise objectives.

Ethics: Use of information, systems, and networks across an enterprise and by all stakeholders matches expectations established by social norms, obligations, being a responsible internet citizen, and enterprise codes of ethical conduct. Policies describing the ethical use of information address ownership, privacy, and prohibition of inappropriate use of information and systems to the detriment of an enterprise and its stakeholders. As a result, stakeholders are educated and thus respect the legitimate interests of others, understanding the extent to which their action or inaction may harm others, and the consequences of unethical behavior.

Inclusion: The perspective and requirements of all stakeholders are represented and considered in forming an enterprise security strategy and program. This includes an appropriate level of stakeholder involvement in the development and review of principles, policies, procedures, processes, and controls. Inclusion can be achieved through a range of communication and elicitation mechanisms such as web sites, newsletters, regional meetings and conferences, and working groups.

Individual Equity: Leaders implement enterprise security "in a manner consistent with the values of a democratic society including the freedom to exchange thoughts and ideas, the free flow of information, the confidentiality of information and communication, the appropriate protection of personal information, openness, and transparency" [OECD 02]. Qualified further, enterprise security "actions do not infringe

upon the obligations, rights and needs of legitimate users when exercised within the legitimate parameters of the mission objectives" [ISSA 04].

Information Sharing: In response to the need for greater transparency and visibility, leaders are prepared to report the organization's security state to stakeholders when and where required, appropriately balanced with the risks of such disclosure. This includes ensuring that the right information is collected, retained, and communicated to the right parties at the right time. Forums for information sharing include those mentioned in Inclusion (above) as well as working with oversight, regulatory, and law enforcement agencies.

Measurement: Leaders articulate metrics that demonstrate the adequacy (or lack thereof) of enterprise security and the extent to which enterprise security actions are aligned with enterprise objectives. Such metrics indicate what leaders consider to be important. "What gets measured gets done. Metrics are about transforming policy into action and measuring performance. Visible metric scores provide a positive influence on human behavior by invoking the desire to succeed and compare favorably with one's peers. Metrics report how well policies and processes are functioning, and whether or not they are producing desired performance outcomes" [CISWG 04b]. Metrics are defined and regularly reported at the governing body, management, and technical levels of the enterprise. Performance measurement of an enterprise's security state is conducted with the same rigor as for other enterprise business units, functions, and processes.

Perspective/Scope: The perspective, scope, and breadth of security considerations is enterprise-wide. "Security consciousness exists at all levels. Security is a holistic issue, including corporate culture, people, training, processes, and communications (not just technical" concerns) [IIA 01]. A coherent system of integrated security protection strategies (principles, policies, procedures, processes, controls) exists to enact all of the principles described here and to ensure continuity of operations. "Security is a fundamental element of all products, services, systems, and networks" [OECD 02] and is considered at each phase of any development and asset life cycle. Staff and stakeholders understand that security is an essential business requirement and thus a characteristic or attribute of how the organization conducts itself.

Response: All accountable stakeholders act in a timely, coordinated manner to prevent or respond to threats to enterprise security and compromises of enterprise security. Such response requires developing and regularly exercising business continuity, disaster recovery, crisis management, and incident management plans so that the enterprise is adequately prepared in the face of an attack and is able to resume normal operations as quickly as possible.

Risk Management: Leaders continually review, assess, and modify enterprise security protection strategies in response to the dynamically changing risk environment in which they operate. This includes "potential harm that may originate from others or be caused by others" [OECD 02]. Leaders articulate acceptable levels of risk (tolerance, appetite, thresholds, assumptions for same) to enterprise assets based on their value, sensitivity, and criticality. Such levels are examined during regular review and assessment processes.

Costs of compromise (loss, damage, disclosure, denied/interrupted access, costs to reconstitute) are quantified to the extent possible as part of ongoing risk management. Controls are selected to effectively monitor and mitigate risk and their performance is regularly measured and reviewed. Plans for remedial action to address risk mitigation deficiencies are developed and executed following each assessment.

Stakeholder interests are most effectively protected by selecting a broad set of enterprise security principles, interpreting and tailoring these for the enterprise, and ensuring their use and enforcement in the normal course of business. These actions aid in ensuring a culture of security. Ultimately, protecting stakeholder interests is about engendering and preserving trust.

We welcome your critique and feedback on this article and any others in the Governing for Enterprise Security series. Please send your remarks to Julia Allen at jha@cert.org.

References

[ACC 99]

American Chemistry Council. *Responsible Care® Guiding Principles*, 1999.

[ACC 03]

American Chemistry Council. *Responsible Care® Security Code of Management Practices*, 2003.

[BSA 03]
Business Software Alliance. “Information Security Governance: Toward a Framework for Action.” October 2003.

[CGTF 04]
Corporate Governance Task Force. "Information Security Governance: A Call to Action." National Cyber Security Partnership, April 2004.

[CISWG 04a]
Corporate Information Security Working Group. Adam H. Putnam, Chairman; Subcommittee on Technology, Information Policy, Intergovernmental Relations & the Census Government Reform Committee, U.S. House of Representatives. "Report of the Best Practices Subgroup." March 3, 2004.

[CISWG 04b]
Corporate Information Security Working Group. Adam H. Putnam, Chairman; Subcommittee on Technology, Information Policy, Intergovernmental Relations & the Census Government Reform Committee, U.S. House of Representatives. “Report of the Best Practices and Metrics Teams.” November 17, 2004; updated January 10, 2005.

[CMMI 03]
Capability Maturity Model Integration. Carnegie Mellon University, Software Engineering Institute. Also Chrissis, Mary Beth, et al. *CMMI: Guidelines for Process Integration and Product Improvement*. Addison Wesley, 2003.

[CRS 05]
Fischer, Eric. “Creating a National Framework for Cybersecurity: An Analysis of Issues and Options.” Order Code RL32777. Congressional Research Service, Library of Congress, February 22, 2005.

[Geer 04a]
Geer, Daniel E. "Why Information Security Matters." Cutter Consortium *Business-IT Strategies Vol. 7*, No. 3, 2004.

[Geer 04b]
Geer, Daniel E. "Security of Information When Economics Matters." Verdasys, May 2004.

[IIA 01]
The Institute of Internal Auditors et al. "Information Security Governance: What Directors Need to Know." IIA, 2001.

[ISO 00a]
International Standards Organization. ISO 9000:2000 *Quality Management Systems —Fundamentals and Vocabulary; Second edition 2000-12-15*. ISO 9000:2000(E), 2000.

[ISO 00b]
International Standards Organization. ISO/IEC 17799 *Information Technology Code of Practices for Information Security Management, First edition*. ISO/IEC 17799:2000(E). December 2000.

[ISSA 04]
Information Systems Security Association. "Generally Accepted Information Security Principles v3.0." ISSA, 2004.

[ITGI 01]

Information Technology Governance Institute. “Information Security Governance: Guidance for Boards of Directors and Executive Management.” Information Systems Audit and Control Foundation, 2001.

[ITGI 04]
Information Technology Governance Institute. “COBIT Security Baseline: An Information Security Survival Kit.” ITGI, 2004.

[NACD 01]
National Association of Corporate Directors. “Information Security Oversight: Essential Board Practices.” NACD, December 2001.

[NIST 96]
Swanson, Marianne & Guttman, Barbara. “Generally Accepted Principles and Practices for Securing Information Technology Systems” (NIST Special Publication 800-14). National Institute of Standards and Technology, September 1996.

[NIST 04]
Stoneburner, Gary, et al. “Engineering Principles for Information Technology Security (A Baseline for Achieving Security), Revision A” (NIST Special Publication 800-27 Rev A). National Institute of Standards and Technology, June 2004.

[OECD 02]
Organisation for Economic Co-operation and Development. "OECD Guidelines for the Security of Information Systems and Networks: Towards a Culture of Security." OECD, 2002.

[1] http://www.ichnet.org/glossary.htm

About the Author

Julia Allen is a senior member of the technical staff in the Networked Systems Survivability Program at the Software Engineering Institute (SEI), a unit of Carnegie Mellon University in Pittsburgh, Pa. The CERT Coordination Center is also a part of this program.

Allen is engaged in developing and transitioning enterprise security frameworks and executive outreach programs in enterprise security and governance. Prior to this technical assignment, Allen served as acting director of the SEI for an interim period of six months as well as deputy director/chief operating officer for three years. Her degrees include a BSci in computer science (University of Michigan) and an MS in electrical engineering (University of Southern California). She is the author of *The CERT Guide to System and Network Security Practices* (Addison-Wesley, June 2001).

The views expressed in this article are the author's only and do not represent directly or imply any official position or view of the Software Engineering Institute or Carnegie Mellon University. This article is intended to stimulate further discussion about this topic.

The views expressed in this article are the author's only and do not represent directly or imply any official position or view of the Software Engineering Institute or Carnegie Mellon University. This article is intended to stimulate further discussion about this topic.

info@sei.cmu.edu
412-268-5800

Social Security Administration Reaps Rewards of Process Improvement

NEWS AT SEI

This library item is related to the following area(s) of work:

Process Improvement

This article was originally published in News at SEI on: April 1, 2005

Most organizations that employ the Capability Maturity Model approach (Software CMM and CMMI methodologies) use it to assess and improve the maturity of their software and systems engineering processes. Software maturity was, indeed, on the minds of officials at the Social Security Administration (SSA) Office of Systems as it began its software process improvement (SPI) program in the late 1990s. But it was another type of maturity that spurred them to take action immediately: nearly one-third of their workforce was due to retire within five years.

Couple with this turnover the impending explosion in the SSA workload as millions of baby boomers retire, and the SSA was facing a potentially devastating "perfect storm" of conditions—managing the largest workload in the organization's history with a staff that did not have the experience of the departing veterans.

Moving from Heroes to Processes

The Office of Systems manages the development, acquisition, and use of SSA's information-technology resources to support the agency's business functions. The key to setting the office on a steady course to deal with the staff changes and evolving IT initiatives was to have well-documented systems and well-defined processes in place, says Rod Waltersdorff, director of the SPI program. Waltersdorff also directs the Division of Process Engineering, Project, and Customer Service in the Office of Systems.

Waltersdorff describes the pre-SPI climate at the Office of Systems as a "culture of heroes" that always depended on people—not processes—to get things done. Consequently, on every important project, the Office of Systems involved its most knowledgeable, experienced people. In many cases, however, these were the same staff members who were counting the days until retirement. A culture change was needed.

Related Links

News

SEI to Co-Sponsor 26th Annual IEEE Software Technology Conference

SEI Cosponsors Agile for Government Summit

Training

See more related courses »

Events

Team Software Process (TSP) Symposium 2014
Nov 3 - 6

SPI: What, Why, and How

Led by Waltersdorff, the SPI group set out some defining principles—what they hoped to achieve, reasons for change, and the method for improvement—in three areas: process improvement, knowledge management, and customer relationship management.

The SPI group defined process improvement as the ability to develop and implement processes, standards, and architectures that help improve the likelihood of success and reduce the risk to a systems project. Process improvement was necessary because the Systems office could no longer depend on individual heroes for success and because interdependencies in development activity and sponsor involvement require discipline. Waltersdorff summarized the goals of SPI goals this way: “Systems wants to deliver what it promises, when it is promised, with the highest quality at the lowest cost.”

To make this happen, Systems began to work with project managers and teams to define best practices, standards, and architectures; provide process and technical guidance to Systems and non-Systems team members; and advise project leaders on project management techniques.

The SSA approach was to use two Software Engineering Institute (SEI)-related models, the Capability Maturity Model for Software (SW-CMM) and the IDEAL model. IDEAL is an organizational improvement model that serves as a roadmap for initiating, planning, and implementing improvement actions. Named for the five phases it describes—initiating, diagnosing, establishing, acting, and learning—IDEAL supports an infrastructure to guide organizations in planning and implementing an effective SPI program and is the founding strategy employed in delivering many SEI services.

A parallel effort in knowledge management was rolled into the SPI program to document corporate knowledge, transmit legacy knowledge, and exchange knowledge both within the division and with its customers, while a related skills inventory initiative helped the Office of Systems identify upcoming skill gaps. These efforts would ease the anticipated increase in workload of the baby-boom retirement wave, and it would allow Systems to respond better to changing customer-service expectations, which were driven by rapid advances in technology.

Likewise, improving responsiveness to customer needs and accountability to customers became an SPI priority. Project teams in Systems accomplished these goals by implementing a defined IT planning and tracking process, identifying key areas of customer satisfaction, and implementing service and customer-communication improvements.

The Right Tools

Throughout the SPI initiative, SSA developed several tools to improve communications and to document risks and processes.

To track project status and to keep project managers and customers informed, Systems developed an intranet-based management tool called the Vital Signs and Observation Report (VISOR) that is accessible to all Systems project managers, executives, sponsors, and customers.

VISOR uses a dashboard with color-coded blocks to provide at-a-glance indicators of project status. Each project, for example, is measured in categories including scope, plans, resources, and risks. Blue blocks indicate satisfactory status while yellow or red blocks represent potential or definite flags. The status blocks link to additional pages with more detailed information. Using VISOR, both project managers and customers can access status information in real time.

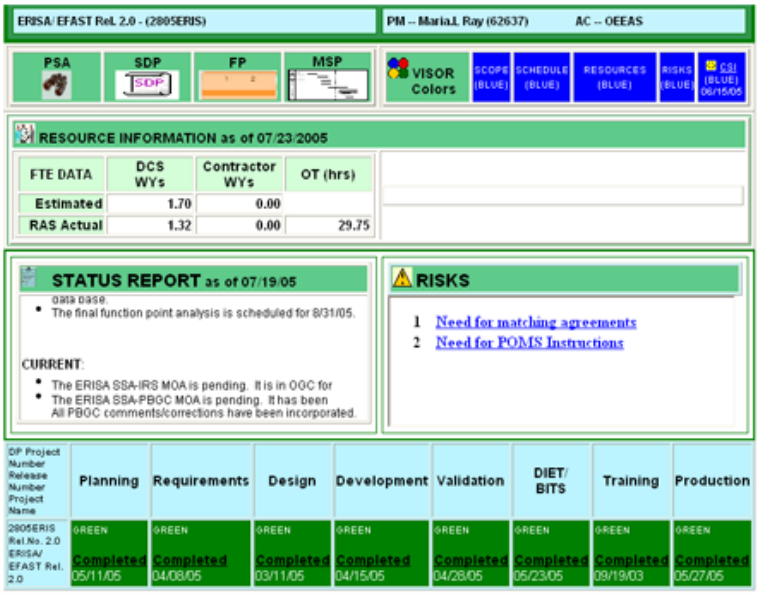

Figure 1: Vital Signs and Observation Report (VISOR)

The SSA Office of Systems developed an intranet-based management tool called the Vital Signs and Observation Report (VISOR), which is accessible to all Systems project managers, executives, sponsors, and customers. VISOR uses a dashboard with color-coded blocks to provide at-a-glance indicators of project status.

For projects that pose a particular challenge or risk, Systems employs another intranet-based tool called Risk Identification and Mitigation System (RIMS) to document risks and guide potential solutions.

The Project Resources Guide (PRIDE), a Web-based tool that defines project lifecycles, provides detailed process documentation for the different types of projects SSA Systems staff may need to plan and execute. Lifecycle categories include Internet, collaboration, and standard projects. The PRIDE documentation guides SSA staff through work requirements, processes, and products. Using the tool during project planning and analysis has helped to control scope creep and requirements volatility and by doing so has produced greater predictability in budget, timeliness, and quality.

The Journey to Maturity

By 2000, the Office of Systems decided to establish a baseline set of core practices that would be followed across the entire development organization and in November 2001, met those goals and achieved Maturity Level 2 of the SW-CMM.

As SSA embraced the Internet as a new and important service delivery channel for the public, Systems staff members realized they needed to develop and employ their most disciplined methods in this area of new technology. Consequently, the Office of Systems Electronic Services (OSES) became the focus of the push to Maturity Level 3.

Steve Kautsch, associate commissioner, OSES, led the effort, which culminated in achieving a Maturity Level 3 rating in February 2005. “We’ve faced enormous challenges over the past five years—building a new organization from the ground up, learning new development tools and technologies, shortening our development timeframes from 12 to 18 months down to 6 months, and growing from 0 to 200 employees,” said Kautsch. “The Office of Systems is often audited to ensure that we follow applicable laws, standards, and best practices, so we cannot cut corners to save time or money. We needed streamlined, efficient processes that manage risk and build in quality. The Office of Systems had selected CMM as the framework, and it helped us define, document, and manage the new processes needed to achieve these goals. OSES began working with the CMM framework in late 2001 and by 2003 had decided to target Maturity Level 3.

“First I made sure every member of the organization understood that process

improvement was a serious commitment from the top," said Kautsch. "I assigned dedicated resources to the effort. We worked closely with the SPI team and the SEI to understand CMM, then established an office-level working group to oversee the development and implementation of the new processes. We worked with all our stakeholders here at SSA to define and document the procedures in a centralized repository for easy access. We trained the staff extensively and enforced compliance using steps that were built into the process. We continually used measurement data such as test results and lessons-learned exercises to refine the procedures. It took three years of focused effort, but it was worth it."

Kautsch cited some of the benefits OSES achieved through process improvement.

- It created standard, well-documented, reusable processes and procedures that reduce the learning curve for new staff and enable staff mobility among projects.
- It streamlined and overlapped tasks that reduced development time by 6 to12 months without sacrificing quality or incurring additional risks.
- It achieved a solid track record based on implementation of more than 50 interactive/transactional public Internet services on time, with the expected scope, and within budget.
- It developed reliable estimates for the cost and schedule of new projects that benefit SSA planners.

Key Findings

The SPI initiative measured key findings from its efforts in five categories: project team characteristics (including evaluations of application experience, language experience, and requirements volatility), productivity, predictability, project estimation, and effort trends. Most measures showed some improvement when comparing 2003 and 2004 results.

Productivity, for example, measured in function points per work month, improved by more than 30%, and the ability to accurately predict the amount of effort to complete a project increased by 12%. Although predictions of project duration based on work months have remained statistically flat, that deviation from the actual duration reported is extremely low at 3.5%. Even so, Waltersdorff interprets this as a success because predictability remained consistent although increasingly volatile projects were being managed by less-experienced staff. In fact, more than 25% of the current Systems staff fall into this category, and he anticipates that the number will rise to more than 30% by the end of the year. Waltersdorff attributes much of the success in these areas to the foundation of defined processes, knowledge management, and attrition planning through Systems' skills-inventory initiative.

Another indication of the program's success is public feedback. Early scores for three of SSA's Internet services from the American Customer Satisfaction Index surveys range from 86-92 and are among the highest scores to date for government or private-sector applications.

SSA continues to refine its data collection and analysis to help gain greater insights into the effects of the SPI initiative.

As the SSA Office of Systems moves forward from its latest appraisal, the SPI initiative will continue the transition from SW-CMM to CMMI. The Office of Systems will also continue to explore its potential role as a contractor for other government organizations; outside customers now include the Centers for Medicare and Medicaid Services, for which SSA is developing an application to manage the recently approved Medicare prescription drug program.

In addition to the measurable data, Waltersdorff sees many intangible results from the intensive SPI effort. "There has been a conscious, noticeable change in the fabric of our organization. You can feel and see the difference in the way people act and talk. They have confidence in their ability," he said. "Changes in the organization go well beyond our process improvement. We feel more mature, more disciplined about what we do."

Kautsch agrees. "Staff members—even our new members—are highly trained and have

the tools they need to succeed. They take satisfaction in producing quality services that are easy and convenient for our clients. Every success builds morale within the organization and raises our credibility both inside and outside the SSA. Project sponsors are satisfied because we do not surprise them with last-minute, unexpected delays or scope changes. It's a win-win for everyone involved."

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

New CERT Course and Handbook Detail Electronic Detective Work

NEWS AT SEI

Author

Eric Hayes

This article was originally published in News at SEI on: April 1, 2005

SEI work in computer forensics to help organizations prepare for legal and technical ramifications of computer security incidents.

The Carnegie Mellon Software Engineering Institute and CERT will begin offering a new training course, *Computer Forensics for Technical Staff*, in January 2006.

The use of the term *computer forensics* and its practice is a recent development in the larger field of computer incident response. The field received governmental funding and recognition when CERT was created in response to the November 1988 "Morris" worm incident, which disabled 10% of the systems connected to the Internet. The Defense Advanced Research Projects Agency (DARPA) charged the SEI with setting up a center to coordinate communication among experts during security emergencies and to help prevent future incidents.

Since that time, organizations have focused on many aspects of computer security, from preventing such incidents from happening in the first place, to "hardening" systems and analyzing product vulnerabilities. CERT has taken a leading role in these efforts, working to ensure that appropriate technology and systems-management practices are used to resist attacks on networked systems and to limit damage and ensure continuity of critical services in spite of successful attacks, accidents, or failures. The goal of this work is to enhance the survivability of computer networks.

The next step in this evolution is an increasingly intense focus on aspects of computer forensics. US-CERT, the operational arm of the National Cyber Security Division at the Department of Homeland Security, defines this term as "the discipline that combines elements of law and computer science to collect and analyze data from computer systems, networks, wireless communications, and storage devices in a way that is admissible as evidence in a court of law" [US-CERT 05].

(< 5 minute) survey.

The SEI recognizes that adding the ability to practice sound computer forensics will help network administrators, computer security staff, and corporate officials ensure the overall integrity and survivability of their networks' infrastructures. For instance, understanding the legal and technical aspects of computer forensics will help organizations capture vital information if a network is compromised and will help prosecution of the case if the intruder is caught. If computer forensics are ignored or practiced badly, organizations risk destroying vital evidence or having forensic evidence ruled inadmissible in court.

The main technical goal of computer forensics is to identify, collect, preserve, and analyze data in a way that preserves the integrity of the evidence collected so it can be used effectively in a legal case. Those who investigate computers must first understand the kind of potential evidence they are looking for in order to structure their search. Second, the investigator must pick the appropriate tools to use. Files may have been deleted, damaged, or encrypted, and the investigator must be familiar with an array of methods and software to prevent further damage in the recovery process.

System administrators and security personnel must also have a basic understanding of how routine computer and network administrative tasks can affect both the forensic process—the potential admissibility of evidence in court—and the subsequent ability to recover data that may be critical to the identification and analysis of a security incident. Finally, anyone overseeing network security must be aware of the legal implications of forensic activity, and security professionals must consider their policy decisions and technical actions in the context of existing laws.

The SEI has researched best practices and evaluated current information on the subject, and is now working to disseminate both the technical and legal knowledge needed to help organizations learn computer forensics principles and practices. Richard Nolan and other CERT professionals have collaborated to produce the First Responders Guide to Computer Forensics, an extensive guide for those who wish to learn more about this burgeoning field [Nolan 05].

This handbook targets a critical training gap in information security, computer forensics, and incident response: performing basic forensic data collection. The focus is on providing system and network administrators with methodologies, tools, and procedures for applying fundamental computer forensics when collecting data on both a live and a powered-off machine. The handbook should help first responders:

- understand the essential laws that govern their actions
- understand key data types residing on live machines
- evaluate and create a trusted set of tools for the collection of data
- collect, preserve, and protect data from live and shut-down machines
- learn methodologies for collecting information that are forensically sound (i.e., able to withstand the scrutiny of the courts)

There are four modules in the handbook. The first describes cyberlaws and their impact on incident response. The second builds understanding of file systems and outlines a best practice methodology for creating a trusted first responder tool kit for investigating potential incidents. The third reviews some best practices, techniques, and tools for collecting volatile data from live Windows and Linux systems and explains the importance of collecting volatile data before it is lost or changed. The final module reviews techniques for capturing persistent data in a forensically sound manner and describes the location of common persistent data types.

The *Computer Forensics for Technical Staff* training course expands on the information in that document. Technical staff who administer and secure information systems and networks will learn more about U.S. cyberlaws and how they affect information security, building and testing safe tool sets, collecting volatile data, and collecting persistent data. Students will have an opportunity to use Helix, Knoppix-STD, Sleuth Kit/Autopsy, dd, PsTools, and many other forensics tools during class. A final scenario puts students into teams and directs them to determine the nature and extent of a suspicious intrusion-detection system alert within a running networked

environment. In addition, the team is directed to collect relevant host and network information for an internal investigation of a questionable email that was forwarded by a "concerned employee."

The first public offering of the *Computer Forensics for Technical Staff* course will take place January 31–February 2, 2006. To register and to obtain more information, visit the SEI's training pages.

References

[Nolan 05]
Nolan, Richard; O'Sullivan, Colin; Branson, Jake; & Waits, Cal. First Responders Guide to Computer Forensics (CMU/SEI-2005-HB-001). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2005

[US-CERT 05]
US-CERT. Computer Forensics (2005).

FAQs Part 2: Are Product Lines Right for My Organization?

NEWS AT SEI

Author

Paul C. Clements

This library item is related to the following area(s) of work:

Software Product Lines

This article was originally published in News at SEI on: April 1, 2005

Leveraging core assets (software, designs, documentation, test artifacts, budgets and schedules, tools, and more) across a family of systems—instead of building each system separately—enables organizations to dramatically increase quality and reduce cost and time to market. But adopting a product line approach to software involves technical and business challenges. This is the second in a series of columns to answer some of the most frequently asked questions of organizations considering a product line approach.

"My organization is very small. Can we build a software product line?"

Of course. There is no reason a one-person software boutique cannot benefit from product line practices. Small organizations would be best served by lightweight versions of the practices. And some of the organizational and role changes that come with product lines may, in fact, be easier to carry out in a smaller organization. Case studies are emerging in which small start-up companies have understood that in, to field a number of different systems, their severe resource constraints compelled them to exploit all possible commonality among the systems. They used a product line approach to leverage their small staffs and budgets across a successful blend of products. An example of such a case study is documented at Salion, Inc. [Clements 02a].

"My organization is very large. Can we build a software product line?"

Of course, and there are many examples of success. The larger the organization, the

Related Links

Training

Software Product Lines - eLearning

Software Product Lines

See more related courses »

more important it is to gain control over the costs and procedures for doing business and to effectively manage products. We have documented case studies of successful software product lines in large organizations also [Brownsword 96].

"How can I make my organization build software product lines when people are busy in their own stovepipes? I don't have enough influence to change the way the organization does business."

Nobody said it would be easy. It is necessary to make the business case for product lines and to find a champion above the level of all the stovepipe organizations. One of the primary missions of the Software Engineering Institute's Product Line Practice initiative is to provide the rationale needed to make the business case to potential champions. Having documented case studies and practices defined within the context of a well-defined framework makes the sales job somewhat easier. Also, there are many adoption strategies open to you. Find a part of the organization where people are open to cross-unit cooperation and begin building a small product line there. Incrementally increase the scope of the product line, and make sure you collect data to record the economies that you gain. The "Launching and Institutionalizing" practice area is especially relevant here.

"The Capability Maturity Model for Software V2 framework puts important aspects of product line practice at Level 4. Do I have to be at Level 4 before I can hope to field a software product line?"

No. But certain Level 2 and 3 practices are key, and successful product line practice does require a certain comfort level with process discipline. For example, an organization must have well-developed configuration-management and product-planning skills before it can successfully field a product line. Organizations that are at Level 4 can transition to product line practices with fewer risks, but the risks can be managed at lower maturity levels as long as they are identified and mitigated.

"What are the various economic motivations for a software product line?"

Many organizations that have started software product lines in recent years have done so out of economic necessity. They have found that they simply cannot continue to do business the old way and still be competitive. Chief among the economic benefits are reduced time to market, opportunities for mass customization, and lower unit costs. These factors are driven by greater productivity from scarce worker resources. Many organizations are finding that they simply cannot find enough qualified people to expand their business without embracing product line practices. In the area of acquisition, the government may commission a product line to eliminate wasteful duplication among programs or to enable it to assemble more cost-effective systems by more easily obtaining products from a range of vendors. See the "Building a Business Case" practice area for more information.

"How long before the approach pays off?"

That's a hard question to answer because it depends on too many organization-specific conditions. If you have decided to build products for which there is no market, then no matter how efficiently you build them, you will not make money. However, a slightly different form of the question can be answered: How many products are required before building them as a product line is more cost-effective than building them as separate systems? Here, emerging data shows that the answer is in the neighborhood of two or three systems. That is, if your product set is expected to be populated by three or more systems, then you're almost certainly better off to build them as a software product line than as separate systems. (See "It Takes Two," [Clements 02b, p. 226].)

"We're doing okay. Why should I change the way I do business and undergo the upheaval that a product line will bring?"

First of all, it's not a foregone conclusion that product line practice brings upheaval. Some adoption strategies prescribe starting small and growing incrementally. New technologies are emerging that are allowing companies to quickly extract and manage commonality among systems they've already fielded; these systems can form the foundation for a product line that can be expanded proactively over time. But no

organization should begin to build a product line without understanding the costs and benefits. Realize that circumstances that motivate an organization are often based on long-term vision and not the status quo. You should ask yourself whether your competition is standing still. The choice to adopt product line practices should be a strategy to achieve specific business goals.

"Can you really be competitive in the marketplace with a software product line? Doesn't it take away your flexibility if you're locked into a product line and, say, an architecture?"

Competitiveness should be sharply increased. What is lost is unbounded variability. However, this is offset by a gain in the flexibility to produce quickly any product in the product line scope. You also gain responsiveness to user needs. The key steps are domain analysis and product line scoping. If domain analysis has captured the requirements of the application domain, the product line has been scoped commensurately, and the architecture reflects those needs, then new products can be brought to market far faster than before with minimal loss of flexibility in comparison to one-of-a-kind systems. However, a product line organization must beware of complacency and should always keep an eye on the emergence of new technologies, new user needs, and new opportunities that might compel a shift in the product line's scope to keep it vigorous into the future.

References

[Brownsword 96]
Brownsword, Lisa & Clements, Paul. A Case Study in Successful Product Line Development (CMU/SEI-96-TR-016). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002.

[Clements 02a]
Clements, Paul C. & Northrop, Linda M. Salion, Inc.: A Software Product Line Case Study (CMU/SEI-2002-TR-038). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002.

[Clements 02b]
Clements, Paul C. & Northrop, L. *Software Product Lines: Practices and Patterns*. Boston, MA: Addison-Wesley, 2002.

About the Author

Paul Clements is a senior member of the technical staff at the SEI, where he has worked for 10 years leading or co-leading projects in software product line engineering and software architecture design, documentation, and analysis. Clements is the co-author of three practitioner-oriented books about software architecture: *Software Architecture in Practice* (1998; second edition, 2003), *Evaluating Software Architectures: Methods and Case Studies* (2001), and *Documenting Software Architectures: View and Beyond* (2002). He also co-wrote *Software Product Lines: Practices and Patterns* (2001), and was co-author and editor of *Constructing Superior Software* (1999). In addition, Clements has also written dozens of papers in software engineering reflecting his long-standing interest in the design and specification of challenging software systems. He received a BS in mathematical sciences in 1977 and an MS in computer science in 1980, both from the University of North Carolina at Chapel Hill. He received a PhD in computer sciences from the University of Texas at Austin in 1994.

The views expressed in this article are the author's only and do not represent directly or imply any official position or view of the Software Engineering Institute or Carnegie Mellon University. This article is intended to stimulate further discussion about this topic.

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

Components of Software Architecture Design and Analysis, The

NEWS AT SEI

Authors

Len Bass

Rick Kazman

Mark H. Klein

This library item is related to the following area(s) of work:

Software Architecture

This article was originally published in News at SEI on: April 1, 2005

The importance of the right software architecture to a development effort has become widely recognized. This trend is probably not surprising to most readers of this column. Consequently this might be an odd time and place to ask why. Why is software architecture a critical software artifact?

The simple answer is that software architecture is important by *definition*. That is to say, software architecture was invented to be an artifact

- defined in terms of elements whose grain is sufficiently *coarse* that the overall design of relatively large systems can be represented in a human-comprehensible form and consequently could aid communication about the overall design of the system; and
- whose specification is sufficiently *detailed* to reason about relative to the satisfaction of critical system requirements

Based on this simple observation, we can state a core principle of software architecture.

- **Principle 1:** A software architecture should be defined in terms of elements that are coarse enough to allow for human intellectual control and specific enough to

Related Links

News

SATURN Conference Announces Additional Keynote, Conference Scholarships

Distinguished Speakers, Strong Technical Program Set for SATURN 2014

See more related news »

Training

Big Data - Architectures and Technologies

Documenting Software Architectures - eLearning

See more related courses »

allow for meaningful reasoning.

Principle 1 alone, however, is not sufficient to reap the potential benefits of software architecture. Principle 1 helps to make the software architecture right. Without understanding, in addition, how to "make the right software architecture," we fall short.

At the Software Engineering Institute (SEI), we have been working on software architecture-related methods and tools for more than a decade and have considerable experience in applying architecture-based design and analysis methods to real systems from a wide variety of domains. From this experience, we have come to believe that there are two other essential principles for realizing the potential benefits of software architecture:

- **Principle 2:** Business (and/or mission) goals determine quality-attribute requirements.
- **Principle 3:** Quality-attribute requirements guide the design and analysis of software architectures.

A software architecture lies at the fulcrum between a system's business (and/or mission) goals and its implementation. Principle 2 states that business goals provide the *raison d'être* for the system. Business goals naturally lead to quality-attribute goals, which, as stated by Principle 3, provide analytic rationale for an architecture to exhibit one type of design versus another.

Together, these three principles allow us to understand why software architecture is important, what purpose it is intended to fulfill, and whether it fulfills that purpose. These principles have been realized in several of our architecture analysis and design methods: the SAAM [Kazman 94], the QAW [Barbacci 03], the ATAM [Kazman 99], the ADD method [Clements 02], and the CBAM [Kazman 01]. More recently, however, we have begun to explore the techniques that link these methods to the principles mentioned above. This allows us to combine these techniques in new ways and create new methods for particular contexts.

The SAAM, the ATAM, and the CBAM work by explicitly identifying the business goals or context, capturing evaluation criteria by scenarios, choosing among criteria based on active stakeholder participation, and relying on the architect for an explanation of how the architecture satisfies the criteria. These methods are effective at identifying which portion of the architecture to examine to determine whether the criteria are satisfied.

These methods use a number of important common techniques.

1. The *explicit elicitation of business goals*. Our methods all have a step that requires a presentation of business goals. This enables external evaluators to determine the criteria with which to evaluate a system. The output of the methods interprets architectural decisions in terms of their impact on the business goals. This provides a means for communicating to management the business impact of technical decisions.
2. Active *stakeholder participation and communication*. We have found that stakeholder concerns are not always expressed in documents and not always well understood by development teams. We include stakeholders in our methods and ensure that they participate in setting of priorities involving the business goals and in setting the focus of the methods. We have developed techniques, such as the "utility tree" [Clements 02], to aid in structured scenario elicitation and prioritization.
3. The explicit *elicitation of architecture documentation and rationale* in standardized views [Clements 03]. To evaluate an architecture, it is necessary for the architecture to be unambiguously represented and clearly understood by the evaluators. Because software architecture is, as we have identified in Principle 1, defined to be at the level of granularity that enables human comprehension, we require that there be such a representation.
4. The use of *quality-attribute scenarios* to characterize stakeholder concerns. Business goals can be expressed at different levels of abstraction. To evaluate a

design, the business goals must be expressed in terms that are operational for the software architect. As we stated in Principle 2, business/mission goals determine the quality-attribute requirements. Quality-attribute requirements must be expressed clearly and unambiguously. We use a specific representation of quality-attribute scenarios called six-part scenarios to express the realization of business goals, and we use general scenarios to aid in the elicitation of these six-part scenarios [Bass 03].

5. The *mapping of quality-attribute scenarios onto the architecture representation* to determine the aspects of the architecture on which to focus. Even though architectures are defined to be understandable, they still may represent large systems and contain much detail. The scenarios are used to focus on particular aspects of the architecture.
6. The representation of *design primitives, called tactics* [Bachmann 03], to make the process of design more consistent and to explicitly link design operations to desired quality-attribute goals.
7. The use of *templates to capture information* and make the methods more consistent among different evaluators. In applying our methods, we have learned that consistency in the execution of a method can be achieved only if there are templates for recording the elicited information and the analyses generated. Templates provide a consistency to the gathering and reporting of information that is useful both for the evaluator and the consumer of the evaluation.
8. The explicit *elicitation of costs and benefits* associated with architectural decisions [Kazman 01]. To rank and make architecture-improvement decisions, it is necessary to elicit information about costs, benefits, and schedule implications of architectural decisions, since these concerns always trade off with pure quality-attribute concerns.

These techniques are efficient at helping designers and analysts find the correct location to examine in an architecture to determine whether a scenario can be achieved. But these techniques provide little support for determining *what* to examine at that location. Most architecture-based methods, ours included, have relied heavily on the expertise of the designers and evaluators when examining the relevant portions of the architecture. Expert opinion has typically been required to determine whether the architectures are satisfactory.

To address this shortcoming, we have recently added two new techniques to the list above:

9. Architectures can be analyzed through the use of *quality-attribute models*. Some quality attributes, such as performance, have well-known analysis models. Other quality attributes, such as variability, testability, and security, have less mature models. In each case, however, we can use the quality-attribute models that exist to help us understand the design decisions made in the architecture.
10. Quality-attribute models lead to a set of *quality-attribute design principles*. Given a particular problem identified by an analysis, there must be some method to generate alternatives for improvement. We have identified a set of design principles based on quality-attribute models, and these principles aid in identifying alternatives.

We are now in the process of creating a follow-on method to the ATAM that combines these techniques in new ways. Stay tuned to subsequent columns for more information.

References

[Bachmann 03]
Bachmann, Felix; Bass, Len; Klein, Mark. Deriving Architectural Tactics: A Step Toward Methodical Architectural Design (CMU/SEI-2003-TR-004). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003.

[Bass 03]
Bass, L.; Clements, P.; Kazman, R.; *Software Architecture in Practice, 2nd edition*, Boston, MA: Addison-Wesley, 2003.

[Barbacci 03]
Barbacci, Mario R.; Ellison, Robert; Lattanze, Anthony J.; Stafford, Judith A.; Weinstock, Charles B.; Wood, William G. Quality Attribute Workshops, Third Edition (CMU/SEI-2003-TR-016). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003.

[Clements 02]
Clements, P.; Kazman, R.; Klein, M. *Evaluating Software Architectures: Methods and Case Studies*. Boston, MA: Addison-Wesley, 2002.

[Clements 03]
Clements, P.; Bachmann, F.; Bass, L.; Garlan, D.; Ivers, J.; Little, R.; Nord, R.; Stafford, J. *Documenting Software Architectures: Views and Beyond*. Boston, MA: Addison-Wesley, 2003.

[Kazman 01]
Kazman, R.; Asundi, J.; Klein, M. "Quantifying the Costs and Benefits of Architectural Decisions," *Proceedings of the 23rd International Conference on Software Engineering (ICSE 23*), (Toronto, Canada), May 2001, 297-306.

[Kazman 99]
Kazman, R.; Barbacci, M.; Klein, M.; Carriere, S.; Woods, S. "Experience with Performing Architecture Tradeoff Analysis", *Proceedings of the 21st International Conference on Software Engineering (ICSE 21*), (Los Angeles, CA), May 1999, 54-63.

[Kazman 94]
Kazman, R.; Abowd, G.; Bass, L.; & Webb, M. "SAAM: A Method for Analyzing the Properties of Software Architectures," 81-90. *Proceedings of the 16th International Conference on Software Engineering. Sorrento*, Italy, May 16-21, 1994. Los Alamitos, CA: IEEE Computer Society, 1994.

About the Authors

Rick Kazman is a senior member of the technical staff at the SEI, where he is a technical lead in the Architecture Tradeoff Analysis Initiative. He is also an adjunct professor at the Universities of Waterloo and Toronto. His primary research interests within software engineering are software architecture, design tools, and software visualization. He is the author of more than 50 papers and co-author of several books, including a book recently published by Addison-Wesley titled *Software Architecture in Practice*. Kazman received a BA and MMath from the University of Waterloo, an MA from York University, and a PhD from Carnegie Mellon University.

Len Bass is a senior member of the technical staff at the Software Engineering Institute (SEI) and participates in the High Dependability Computing Program. He has written two award-winning books in software architecture as well as several other books and numerous papers in a wide variety of areas of computer science and software engineering. He is currently working on techniques for the methodical design of software architectures and to understand how to support usability through software architecture. He has been involved in the development of numerous production or research software systems ranging from operating systems to database management systems to automotive systems.

Mark Klein is a senior member of the technical staff of the Software Engineering Institute. He has more than 20 years of experience in research on various facets of software engineering, dependable real-time systems and numerical methods. Klein's most recent work focuses on the analysis of software architectures, architecture tradeoff analysis, attribute-driven architectural design and scheduling theory. Klein's work in real-time systems involved the development of rate monotonic analysis (RMA), the extension of the theoretical basis for RMA, and its application to realistic systems. Klein's earliest work involved research in high-order finite element methods for solving fluid flow equations arising in oil reservoir simulation. He is the co-author of two books: *A Practitioner's Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems* and *Evaluating Software Architecture: Methods and Case Studies*.

Find Us Here

Share This Page

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

Introducing a Guide to Interoperability

NEWS AT SEI

Author

Dennis B. Smith

This library item is related to the following area(s) of work:

System of Systems

This article was originally published in News at SEI on: April 1, 2005

Within the Department of Defense (DoD), government, and corporate worlds, large-scale systems of systems (SoS) are increasingly being put together in an unprecedented way. Establishing interoperability between the constituent systems is increasingly a key for an organization to meet its critical goals. However, as organizations envision transparent net-centric operations spanning a large number of systems, we have found that successfully achieving such a vision requires a fundamental shift in traditional ways of thinking about system development, acquisition, and management.

In a seminal contribution Mark Maier has argued that systems of systems have unique characteristics that make them fundamentally different from individual systems [Maier 96]. These are

- Operational independence of the systems: Each of the individual systems within an SoS has a life of its own and can function acceptably without necessarily interacting with other systems.
- Managerial independence: The individual systems within an SoS are controlled by different authorities.
- Evolutionary development: Each system within the SoS is developed and upgraded according to a different schedule.
- Emergent behavior: The behavior of the SoS as a whole is not embodied in any one of the systems within the SoS.
- Geographic distribution: The systems within the SoS are not necessarily co-located.

Related Links

Training

Migrating Legacy Systems to SOA Environments - eLearning

Service-Oriented Architecture: Best Practices for Successful Adoption

See more related courses »

We have found that, consistent with this overall formulation, systems of systems display a global complexity that cannot be managed by hierarchical structures and central control. In fact, the application of traditional approaches to system and software development may result in the development of rigid stovepipes that are ill-suited to the needs of net-centric world of loosely coupled interoperating components.

Despite the critical need for understanding how to make systems interoperable within an SoS context, there has not yet been a guide for understanding what interoperability is, when it needs to be applied, and how to effectively achieve it. In fact, a common view is that interoperability equates to connectivity and applying the right standards. While standards are necessary, they are far from sufficient.

As we develop our understanding of some of these issues, we are creating a web-based Guide to Interoperability. This *Guide*, over a period of time, will attempt to fill that gap as an evolving, web-based document intended to assist those who are planning, constructing and maintaining interoperable systems of systems. The *Guide* provides an outline of the organization, technology, and engineering issues that need to be addressed to achieve effective interoperability. It captures best practice and points to promising approaches for construction of highly interoperable systems of systems. Currently, initial drafts exist for a relatively small number of topic areas. The rest of the areas will be filled in over time, and the current drafts will be updated as new information is assimilated.

The *Guide* has four sections: Introduction, Engineering Practices, Management Practices, and Technologies. This recognizes that the problem of interoperability requires that each of these areas needs to be addressed.

The current outline of the *Guide* is provided below, together with a brief summary of its content. Initial drafts currently exist for the underlined sections.

I. Introduction
A. The SoS Interoperability Problem will outline the basic issues of interoperability within an SoS context, including emergent properties, independence of component systems, and independent evolution.
B. Characteristics of Interoperability will describe the characteristics of interoperability, such as ownership issues, semantic consistency, evolution, dynamism, data management, and communication.
C. Models of Interoperability outlines current models that have been proposed to understand interoperability, including levels of information systems interoperability (LISI), levels of conceptual interoperability model (LCIM), layers of coalition interoperability (LCI), and system of systems interoperability model (SOSI).

II. Engineering Practices
A. Establishing a Concept of Operations
B. Identifying Interoperability Requirements
C. Defining Architectures for Interoperability identifies the mechanisms used by architects to accommodate future change with minimal impact, such as specialized layering, high modularity, well-defined interfaces, standard interfaces, and common integration mechanisms. Architectures that have been used or proposed for achieving interoperability are discussed, including serviced-oriented architectures (SOAs), grid architectures, and component-based architectures.
D. Designing for Interoperability
E. Procuring Interoperable Components discusses approaches for procuring components for systems of systems, including COTS component evaluation and methods for legacy component reuse evaluation.
F. Integrating the SoS discusses mechanisms for achieving integration—including specialized architectures such as SOAs, databases integration, integration through published application programming interfaces (APIs), file sharing, and memory mechanisms and queuing—and outlines classic and updated approaches for dealing with architectural mismatches.
G. Testing Interoperability
H. Maintaining Interoperability
I. The Future of Engineering Processes for SoS

III. Management Practices
A. Building the Business Case for Interoperability outlines issues that motivate SoS interoperability, including faster response to market demands, better decision making, cheaper adoption of new strategies, and adaptive service-oriented networks.
B. Developing an Acquisition Strategy outlines a set of acquisition-related strategies that are being incorporated within the U.S. DoD to facilitate SoS acquisition and identifies groups that are addressing issues of interoperability such as the Open Systems Joint Task Force (OSJTFE), and World Wide Web Consortium (W3C).
C. Establishing a Software Development Lifecycle
D. Planning
E. Contract Management
F. Risk Management
G. Configuration and Change Management
H. Monitoring Progress
I. Planning for Sustainment
J. Managing the Organization

IV. Technology
A. Technology Evaluation identifies approaches for technology evaluation, including COTS evaluation technologies, evolutionary testbeds, and a model problem approach for hands-on evaluation.
B. Technologies for Interoperability currently has subsections that summarize the basic concepts and approaches, state of the practice, and implications for systems of systems for each of the following:
a. Web Services
b. Component Frameworks
c. Model Driven Architecture (MDA)
d. Ontology Web Language for Services (OWL-S)
e. Open Grid Services Architecture (OGSA)

As a web-based document, the *Guide* will evolve over the next several years. The outline provides our initial view of the primary interoperability issues. This outline will be revised as we and the rest of the community gain additional experience.

We solicit your comments and involvement. Rather than waiting several years to present a more complete guide, we have decided to provide releases as a work in progress. New versions of the *Guide* will be released regularly to update and improve the information provided.

We encourage you to become part of this effort to improve the practice of building interoperable systems by providing feedback on these pages and by sharing your experiences and ideas. Please mail comments to the Integration of Software Intensive Systems (ISIS) Initiative at isis-sei@sei.cmu.edu. We will consider your comments in future updates and hope that it will begin to provide useful information over the near term.

1 Interoperability is the ability of a collection of communicating entities to share specified information and to operate on that information according to a shared operational semantics in order to achieve a specified purpose in a given context.

Reference

[Maier 96]
Maier, Mark, W. Architecting Principles for Systems-of-Systems. http://www.infoed.com/Open/PAPERS/systems.htm 1996.

About the Author

Dennis Smith is the lead for the SEI Integration of Software Intensive Systems Initiative. This initiative focuses on addressing issues of interoperability and integration in large-scale systems and systems of systems. Earlier, he was the technical lead in the effort for migrating legacy systems to product lines. In this role he developed the method Options Analysis for Reengineering (OARS) to support reuse decision-making. Smith has also been the project leader for the CASE environments project. This project examined the underlying issues of CASE integration, process

support for environments and the adoption of technology. Smith has published a wide variety of articles and technical reports, and has given talks and keynotes at a number of conferences and workshops. He has an MA and PhD from Princeton University, and a BA from Columbia University.

The views expressed in this article are the author's only and do not represent directly or imply any official position or view of the Software Engineering Institute or Carnegie Mellon University. This article is intended to stimulate further discussion about this topic.

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

Large-Scale Work—Part III: The People

NEWS AT SEI

Author

Watts S. Humphrey

This library item is related to the following area(s) of work:

Process Improvement

This article was originally published in News at SEI on: April 1, 2005

This is the third column in the series on large-scale development work. The first column briefly summarized the issues of large-scale development work, addressed the problems of organizational size, and described how organizations necessarily become less efficient as they grow. The second column dealt with project issues, particularly with the management-decision process. Often, the most serious problems in large projects concern the way decisions are made. In this column, I continue this discussion with a principal focus on the people and teams that make up the project staff and the issues involved in maximizing their performance.

Yet Another War Story

While I was still at IBM, Peggy, a project manager, came to me for advice on how to handle a disagreement with marketing over her planned product. She had worked in my group a few years before and knew that I had dealt with such problems many times. She wanted my advice on how to proceed. Marketing staff insisted she add some functions that she did not believe were worthwhile. She debated this issue with the marketing representatives, and they were adamant. However, she had also talked to IBM's GUIDE user group about the issue, as well as to several customers who were members of her user advisory board, and none of them felt that the functions were important enough to warrant delaying the product's availability. They wanted the product as soon as they could get it, and these added functions would cause a delay of several months.

Marketing still held out. I suggested that Peggy inform marketing that she was proceeding without their approval and that she write a note to her management and copy marketing management explaining her position. I told her that the marketing

Related Links

News

SEI to Co-Sponsor 26th Annual IEEE Software Technology Conference

SEI Cosponsors Agile for Government Summit

Training

See more related courses »

Events

Team Software Process (TSP) Symposium 2014

Nov 3 - 6

people would be surprised and that they wouldn't know what to do. This was not a sufficiently important issue to involve senior management, so if she were firm and had a sound business and technical position, marketing would cave in. That is what she did, and marketing did cave in.

Bureaucratic Blackmail

As organizations grow, they become increasingly bureaucratic: staffs and review departments are established to monitor the operating groups. The larger organization also needs larger bureaucratic functions to handle the organization's routine activities like delivering the mail, handling the payroll, running the facilities, and obtaining and distributing supplies. As the size of these staffs increases, their efficiency and effectiveness become a major concern, and management is likely to establish even more bureaucracy to monitor, measure, and track the growing bureaucracy.

So, bureaucracies grow seemingly without bound. Every new problem causes a reaction that often adds to the bureaucracy and makes organizations less responsive, more rigid, and less efficient than before. Of course, some bureaucracy is essential, and bureaucratic procedures can often be helpful. There are lots of routine things that must be done in any large organization, and neither management nor development groups want to worry about them. With an administrative function to handle them, these routine things become automatic.

Once these functions become automatic, however, they also become rigid and hard to change. Therefore, since development's job is to develop new things, which invariably requires some degree of change, and since the bureaucracy's job is to resist change, conflict is inevitable. This means that in large organizations, to get anything done, development groups must know how to overcome bureaucratic resistance. This task is not trivial because, once these bureaucratic groups get power, they tend to use that power in ways that management had not intended.

IBM management did not want the marketing group dictating to the project manager which functions should be included in her product. While marketing should have a voice, and development groups should take advantage of their guidance, product managers must be responsible for the performance of their products. This means that they must have the final say on what goes into their products, how the company's money is spent, and how their products are introduced and offered.

While Peggy's experience would seem to put the marketing people in a bad light, this is just an example of a staff group that overreached. There are many other cases in which such groups properly use their sign-off authority and help product managers produce truly superior products. There are also other cases in which staff groups identify real product problems and help to avert real mistakes.

Perhaps the worst case I know of in which a group used bureaucratic procedures to force a technical position happened some years ago when IBM was considering developing a new low-cost digital fax machine. The initial product concept was for a simple machine that would produce high-quality fax output. At the time, there was no comparable machine available. The marketing people kept pushing for additional functions, and the product manager kept caving in. Finally, after many months, the planned product's costs had grown so much that the product could no longer meet the target price for a low-end machine. By the time the product was replanned for the proper market niche, a competitive machine had been introduced, and the market opportunity was lost.

If this product manager had fought for the original low-end product concept, IBM would have had an attractive new product, well ahead of the competition. So the key question is, "How can a developer operate in a large and increasingly rigid bureaucratic morass?" As the size and power of these bureaucratic groups increase, it seems that everybody is trying to block the developer's work. Everyone can say "no," and, seemingly, no one can say "yes." It's like "death by a thousand cuts." If you spend your life fighting the bureaucracy, you won't have time to do anything else. How can developers get anything done? There are a few guidelines that can help.

First, don't sweat the small stuff. While none of us likes inefficiency and seemingly-

stupid bureaucratic rules and regulations, assume that this stuff is there for a reason, and as long as it doesn't block your ability to do your job, or if you can get around it without too much pain, learn to ignore it. Somebody else is worrying about organizational efficiency, so concentrate on your job of developing products.

In one example of this problem, Bret had just been made manager of a project to build a new communication device. As he talked to the developers, one of the new engineers complained about the stock room. It never had the right parts in stock, and the engineers often had to wait weeks for the parts they needed. Bret thought that this should be an easy problem to solve, so he talked to the stock-room manager. He found that there were all kinds of outdated rules governing stocking levels, reorder procedures, and purchase authorizations. After several days of digging, he found that, to change the system, he would need the lab director's approval. He wisely decided not to pursue the matter.

Second, if the bureaucracy gets in the way, try to think of how you could do your job in a way that would not cause those bureaucratic problems. We often have choices in doing our work, and if some simple adjustments will suffice, make the changes and don't fight over them. Before you go into battle, make sure there is no simple procedural fix that will do the job.

Instead of going to the lab director, Bret discussed the situation with Pete, a seasoned technician. When he told Pete about the problem, Pete asked: "What do you need?" Bret told him, and the next morning, Pete walked into his office with the parts. When Bret asked where he got them, Pete described the "midnight requisition" system that the technicians had developed to get scarce parts. Because the stock room was so antiquated, the technicians on all the projects had gradually accumulated enough private stock of frequently needed parts to meet their projects' emergency needs. By checking with his buddies, Pete quickly got the parts he needed.

Third, when the bureaucracy really does get in your way, you will have to do battle. When you do, focus on getting your job done. In these fights, remember that the bureaucracy has one strong point—procedures. You are doing battle because what you want to do doesn't fit their established procedures, and you will always lose battles over procedures. So don't propose changing procedures or even new ways of interpreting procedures. Your strong points are business oriented: customer responsiveness, product enhancements, or other business needs that you are charged with addressing. As you escalate issues, your position should be, "I don't understand your procedural problem; all I know is that we have to address this business problem, and I need your help in figuring out how to do that."

Some weeks later, while testing the first system prototype model, Bret's team had a disaster and burned out several critical parts. A customer demo was scheduled in a couple of weeks, and the stock room said it would take a month to get the needed parts. Pete's midnight requisition system couldn't help but he did locate a supply of the parts on eBay. Since the stock room had no way to buy parts on eBay, however, Bret was stymied. This time, he went to the lab director and got the authorization to buy the parts on eBay.

You may have to keep escalating a bureaucratic problem until you reach an executive who is responsible for both bureaucratic and business issues. But once you do, if you have a clear and convincing business story, that executive will always support you. Then you will get the help you need, and the bureaucrats will be directed to help you.

In the next column, I continue this discussion of large-scale work with a special focus on the politics of organizations and why democratic principles are particularly hard to apply in large development groups.

Acknowledgments

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

In Closing, an Invitation to Readers

In this particular series of columns, I discuss some of the development issues related to large-scale projects. Since this is an enormous subject, I cannot hope to be comprehensive, but I do want to address the issues that you feel strongly about. So, if there are aspects of this subject that you feel are particularly important and would like covered, please drop me a note with your comments, questions, or suggestions. Better yet, include a war story or brief anecdote that illustrates your ideas. I will read your notes and consider them when planning future columns.

Thanks for your attention and please stay tuned in.

Watts S. Humphrey
watts@sei.cmu.edu

About the Author

Watts S. Humphrey founded the Software Process Program at the SEI. He is a fellow of the institute and is a research scientist on its staff. From 1959 to 1986, he was associated with IBM Corporation, where he was director of programming quality and process. His publications include many technical papers and several books. His most recent books are *Introduction to the Team Software Process* (2000) and *Winning With Software: An Executive Strategy* (2002). He holds five U.S. patents. He is a member of the Association for Computing Machinery, a fellow of the Institute for Electrical and Electronics Engineers, and a past member of the Malcolm Baldrige National Quality Award Board of Examiners. He holds a BS in physics from the University of Chicago, an MS in physics from the Illinois Institute of Technology, and an MBA from the University of Chicago.

The views expressed in this article are the author's only and do not represent directly or imply any official position or view of the Software Engineering Institute or Carnegie Mellon University. This article is intended to stimulate further discussion about this topic.