CMMI: A New Transition

NEWS AT SEI

Author

Mike Phillips

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

Process Improvement

CMMI

This article was originally published in News at SEI on: January 1, 2007

This is one of those challenging periods in which we make the transition from one version of CMMI to the next and work on new variants to expand CMMI coverage. In this column, I'll cover each element of the CMMI Product Suite, including the models, the appraisal method, and the training elements across the product lifecycle as we phase out Version 1.1, phase in Version 1.2, and prepare products for release in 2007.

First, the Models...

We are phasing out... CMMI-SE/SW/IPPD/SS, V1.1, and its eight model variations. The models have been in use since 2002. The sunset of the model, as defined by the last acceptable appraisal, is scheduled for the end of August 2007. By that time, we expect that approximately 60,000 people will have been trained around the world and about 2,000 appraisals will have been conducted.

We are phasing in... CMMI for Development (CMMI-DEV), V1.2, a single book with an embedded model variant for IPPD. Minor distinctions that characterize the two appraisal options (staged or continuous) are included in the same book. Complexity has been reduced, but some new features have been added. A second edition of the CMMI book, CMMI: Guidelines for Process Integration and Product Improvement, 2nd Edition, published by Addison-Wesley, is available; it includes new features (e.g., hints and tips) not found in the earlier version. The elimination of legacy approaches from source models and the restructuring of IPPD practices have saved about 15% in model size.

We are developing... two new focus areas for CMMI process improvement. An

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

architectural improvement early in the V1.2 development has allowed consideration of these new areas. Both areas are closely related to the development lifecycle. One area includes acquisition processes during development and the other area includes service delivery processes after development is complete. In progress now for release in 2007 are a constellation for acquisition and another for services. Each maintains at least 75% commonality with the development constellation to ease training and appraisals. All of the common elements of these new constellations will be marked to show the full reuse of best practices for project management, process management, and support.

CMMI for Acquisition (CMMI-ACQ) is already available in an initial draft form on the SEI Web site, and piloting has shown some areas that can be improved before the planned release this spring. Work has also begun on a CMMI constellation for service delivery, (CMMI-SVC). Process areas focused on service delivery activities have been initially drafted and will be scheduled for piloting and refinement after the release of CMMI-ACQ.

The expansion of the scope of CMMI has suggested the need for other documents. A master glossary, for example, may be a useful compilation of key terms from the growing number of constellations along with those critical to understanding the SCAMPISM appraisal method. It may also prove useful to publish separately the process improvement foundation material that we are calling the CMMI Model Foundation (CMF), which consists of the model elements common to all CMMI constellations.

Second, Appraisals...

We are phasing out... appraisals based on the use of the SCAMPI V1.1 Method Definition Document and its associated training. This document and training can continue to be used through August 2007 to give sufficient time for current lead appraisers to receive upgrade training and pass an upgrade test. Appraisers can use either Version 1.1 or 1.2 of the model, as determined in appraisal planning with the organization. The result will be characterized in the SEI database as a V1.1 appraisal. The appraisal results will have the same three-year lifetime associated with V1.2. The modifications to the Appraisal Disclosure Statement that resulted from the V1.2 revision and the use of the new SCAMPI Appraisal System for reporting appraisals are both available for either version during the transition period.

We are phasing in... the SCAMPI V1.2 Method Definition Document and its associated training. While the changes to the Method Definition Document are primarily clarifications of elements of the method, these changes are designed to increase confidence in appraisal results. The changes to SCAMPI V1.2 are at least as significant as the model revisions. One of these changes is the need to sample appraisal input from across different projects so that institutionalization across the appraised organizational unit can be assured.

The greater appraisal-related changes relate to authorized lead appraisers. Lead appraisers have at least two and possibly three steps for V1.2 qualification. The first two steps are required for all lead appraisers: (1) attend one of the face-to-face meetings held with the CMMI Appraisal Team at conferences or other locations to assure understanding of the Code of Conduct and expectations for appraisals, and (2) pass a test on the changes and on key areas of concern. Both of these requirements must be met before current V1.1 lead appraisers can conduct a V1.2 SCAMPI appraisal. A third requirement applies to lead appraisers who conduct high-maturity appraisals--those appraisals addressing either maturity or capability levels 4 and 5. Lead appraisers wishing to lead these types of appraisals must also pass an oral exam administered by examiners familiar with the necessary elements of high maturity in organizations. The oral exam enables the SEI to provide an interim certification, but has confirmed the need for a full certification approach for lead appraisers in the future.

We are developing... We are beginning our planned work on a Lead Appraiser Body of Knowledge that will be the basis for continuously improving the competency of those who are lead appraisers for CMMI products. By October, we plan to have a full certification program in place to replace the interim approach we have today. While the

interim approach focuses on high maturity, future certification will apply to all lead appraisers and therefore all aspects of appraisals, not simply the high-maturity elements.

Finally, Training...

We are phasing out... Version 1.1 training courses. A sufficient number of instructors have become qualified to assure that all courses delivered in 2007 and beyond can be V1.2 courses. These new courses reflect the reduction of the model by three process areas, so the courses now more effectively use their allotted time. Explanations about common features and advanced practices are no longer necessary. However, these improvements do create a difference between V1.1 and V1.2 training that must be addressed during the transition period, because some organizations train their appraisal team members immediately before the appraisal. So those organizations performing V1.1 appraisals through August must provide these team members with material that was in V1.1 but not in V1.2. We are working with these organizations to address this transition-specific issue.

We are phasing in... Version 1.2 training courses. Most of the authorized instructors have completed the upgrade, and our Partners have not indicated any difficulties with the transition thus far. With V1.2, we will allow use of either the SEI technical report or the recently released Addison-Wesley textbook. Each choice has its proponents, so we will leave the choice to the Partner delivering the course.

Our experience using Internet-based training courses for transition has built confidence in greater use of these tools in the future. We expect use of the Internet-based CMMI Upgrade course by instructors and lead appraisers will end this summer. We will, however, continue to offer the upgrade course for appraisal team members since we need to assure that experienced V1.1 appraisal team members can be easily upgraded to V1.2 team members without having to retake the Introduction to CMMI course.

We also have converted the Intermediate Concepts of CMMI course and the Instructor Training course to V1.2 capability so that all future candidates can teach the V1.2 material.

We are developing... a strategy for teaching future CMMI courses most effectively. The creation of two new constellations that have significant commonality with CMMI-DEV is causing us to carefully consider the best strategies for providing CMMI courses in the future. Our current thinking is that initially the new constellations will consist of modules that build on those of the existing CMMI-DEV course. However, over time we feel that a restructuring of the existing Introduction to CMMI course material may allow a more tailored approach, substituting modules for development, acquisition, or services as necessary. Final determination of the most effective approach awaits better definition of the two new constellations.

Summary

The transition to V1.2 was more challenging because of the significant increase in users and Partners since the transition from V1.0 to V1.1. Version 1.2 includes a number of improvements across the product suite to make this process-improvement tool more useful and to improve confidence in the results of CMMI-based process improvement. Transition periods always present both known and unexpected challenges. We appreciate how well our Partners and the community of CMMI users have helped us make the transition as rapid and effective as possible.

About the Author

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.

Find Us Here

Share This Page

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

FAQs Part 5: Getting Started

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, 2007

"All right, I want to start a software product line. What do I do first?"

There are many resources available to you, many of which are available on or through our web site. You can

- Read case studies and experience reports of organizations pursuing software product lines. These will help you define some specific goals for adopting the approach and understand what might be involved for you. Assign those studies and reports most applicable to your situation as reading for interested people in your organization.
- Get to know the SEI's *Framework for Software Product Line Practice*, especially the Essential Activities section.
- Learn the process of adopting software product line practices in your organization. You'll need to choose an adoption strategy. The "Launching and Institutionalizing" practice area of *A Framework for Software Product Line Practice* lists several approaches. You can also use product line practice patterns, described in Chapter 7 of *Software Product Lines: Practices and Patterns* [Clements 01], to help with adoption and getting the product line started.
- Sign up for a course on software product lines.
- Start to build a business case for making the switch to software product lines.
- Get involved with the software product line community by participating in one of the conferences and workshops.

Related Links

Training

Software Product Lines - eLearning

Software Product Lines

See more related courses »

Sign up your organization for an SEI Product Line Quick Look or an SEI Product Line Technical Probe to help gauge your organization's strengths and weaknesses with respect to software product line capability. Because the probe relies on interviews with many different people throughout the organization, it's also a good activity to use to begin getting people to think about the product line approach and the organization's goals for adopting it.

Other steps are involved, but these will get you off to a good start.

"My company builds a broadly related group of products, each of which is a group of closely related products. Should I plan to build a single product line, or a group of product lines?"

This can be answered by making a business case for each alternative and weighing the costs and benefits. The SEI's Structured Intuitive Model for Product Line Economics (SIMPLE) can help with this. We've seen this situation in practice many times. Often, the choice has almost always been to go with the single, large product line. Examples come from avionics, missile software, embedded engine controllers, and shipboard command-and-control systems. However, other organizations have chosen to work with separate product lines, or even hierarchical product lines. A hierarchical product line is essentially a product line of product lines. The decision depends on the amount of commonality that can be extracted from the broadly related group, and also how easy it is to communicate and cooperate across the different groups involved. If the products exist in separate business units, for example, the organization might find separate product lines to be more manageable. Finally, if you're just beginning to use the product line concept, it will be much easier to launch one product line than several.

"I want to pilot software product lines in my organization. What are the criteria for a good pilot project?"

The general criteria are the same as for any pilot project. It should be manageable and not too complex. It should be strategic, but not so central that the failure of the pilot will bring down the organization. It should build on strengths rather than weaknesses. Product line pilots should be in an established and well-understood problem area and should be led by some of the best innovators. Starting with a legacy system rather than starting from scratch makes sense if the legacy system is in good health (architecturally sound, well documented, and using modern technologies). If there is an established core asset base from which the product line can be built, so much the better. Finally, the scope of the pilot should be narrow enough that results can be achieved and assessed quickly. See the "Launching and Institutionalizing" practice area for more information.

"Should I plan to take the proactive approach or the reactive approach to my software product line?"

As described in the *Framework* section "All Three Together," the proactive approach involves building the core assets first and then using those to spin out products, whereas the reactive approach calls for having products first and then extracting the core assets from those products. These are two extremes of a spectrum of possibilities in between. A likely compromise is to develop some fresh core assets while producing others from an existing product or stable of products. A key determinant is how well you can predict the future. If you can confidently map out the scope of your software product line in detail, describing the commonalities and variabilities that your products will require, then proactively building the core assets to serve that scope will probably provide a rapid product-production capability. If, on the other hand, your scope is defined only in more general terms, then trying to build the core assets ahead of time will probably result in a large amount of re-work and frustration. In that case, it is better to refine the core assets as you field more products and gain more knowledge about your market. Use whatever information you have when you have it--but no more and no sooner.

"Where is the resistance to adopting a product line approach usually found?"

It can come from almost anywhere, but we observe that often it shows up at the middle-management level. Many times the front-line staff members recognize the

benefits of doing things the new way because they're the ones who must implement and re-implement almost identical applications multiple times. Conversely, senior management tends to have the vision and see the financial bottom lines. This is by no means always the case, however. We also know of an organization in which senior management was preoccupied with buy-outs and mergers, and middle management stepped in and initiated the transition to product line practice; when senior managers turned their attention inward, they discovered to their surprise that their company was building software in an entirely new way.

"Where can I go to get more information?"

The SEI's Product Line Practice initiative Web site is a good place to start.

There are also several books on software product lines, including *Software Product Lines: Practices and Patterns* [Clements 01], *Software Product-Line Engineering: A Family-Based Software Development Process* [Weiss 99], and *Design and Use of Software Architectures: Adopting and Evolving a Product-Line Approach* [Bosch 00]. In addition, *Software Reuse: Architecture, Process, and Organization for Business Success* [Jacobson 97] is oriented toward projects employing large-scale strategic reuse and is compatible in many ways with product line practice. A new book by van der Linden, Schmid, and Rommes called *Software Product Lines in Action* [van der Linden 07] is expected this year.

Conferences on software product lines produce papers about both theory and practice. The premier product line conference is the International Software Product Line Conference (SPLC); the next one will be in Kyoto in September.

Where can I read about other organizations that have successfully adopted the software product line approach?

Case studies are excellent resources to help you get started because they show you how other organizations approached the product line issues they faced.

In addition, you should visit the Software Product Line Hall of Fame, where software product lines of lasting community value are inducted at each Software Product Line Conference. Each inducted product line is described, and references for more information are given.

[Bosch 00]
Bosch, J. *Design and Use of Software Architectures: Adopting and Evolving a Product-Line Approach*. Reading, MA: Addison-Wesley, 2000.

[Clements 01]
Clements, P. & Northrop, L. *Software Product Lines: Practices and Patterns*. Boston, MA: Addison-Wesley, 2002.

[Jacobson 97]
Jacobson, I.; Griss, M.; & Jonsson, P. *Software Reuse: Architecture, Process, and Organization for Business Success*. Reading, MA: Addison-Wesley Longman, 1997.

[Weiss 99]
Weiss, D. M. & Lai, C. T. R. *Software Product-Line Engineering: A Family-Based Software Development Process*. Reading, MA: Addison-Wesley, 1999.

[van der Linden 07]
Van der Linden, F.; Schmid, K.; & Rommes, E. *Software Product Lines in Action*, Springer, 2007, in publication.

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.

Large-Scale Work–Part VI: The Process

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, 2007

This is the sixth column in a series on large-scale development work. The prior columns covered several important preparatory topics, but with this column the focus switches to process issues and the methods for actually doing the work. Part I introduced the structural problems of the massive organizations that typically do large-scale work. Part II addressed the management decision process, why this process is critical for large-scale work, and how to fix it if it is broken. Part III discussed how to get things done in a bureaucratic organization, and Part IV addressed the need for teams to take charge of their own work. Part V then described self-directed teams and how their planning and self-management skills help them to consistently produce superior project results.

The final columns in this series deal with actually doing the work. Once you have a suitable organization structure, management style, and work environment, two challenges remain: how to select the right process and how to get people to follow that process in doing the job. This column addresses the process question.

The Process Problem

The current situation with large-scale system-development processes can be summarized as follows:

- Large and complex computer-based systems are now critical to the economic and military welfare of the United States and to much of the developed world.
- With current methods, development of these systems is typically late and over cost and results in poor-quality products.

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

- Since future development programs will be far more challenging, the methods of the past will produce even worse results and often will not deliver any products at all.
- To address this problem properly, organizations must adopt sound processes that can be scaled up to the larger development challenges of the future.

Selecting a Process

Large and complex computer-based systems are now becoming widely used in the United States and in much of the developed world. If history is any guide, developing these systems with the methods of the past will almost certainly yield unsatisfactory results. These projects have almost always failed because of project-management problems. While the solutions to these problems are known and have proven to be highly effective, they are not yet widely practiced. Therefore, to successfully develop the even larger and more challenging systems of the future, we must start to use processes that address these historical problems. To do this, the processes must meet five requirements.

1. control development cost and schedule predictably
2. handle changing needs responsively
3. minimize the development schedule
4. be scalable
5. produce quality products predictably

Cost and Schedule

To control cost and schedule predictably, organizations must use processes that do five things:

1. Have the people who will do the work estimate and plan that work.
2. Track the progress of the work precisely and regularly.
3. When progress falls behind plan, promptly identify and resolve the causes.
4. When the requirements change, re-estimate and revise the entire plan promptly.
5. Anticipate and manage risks.

Changing Needs

To handle changing needs responsively, it is essential to do the following:

1. Examine every proposed change to understand its implications for the development plan.
2. Pay particular attention to each change's impact on completed work, including requirements, design, implementation, verification, and test.
3. Estimate all of the cost and schedule consequences of making the needed changes.
4. If the cost and schedule implications are significant or if they exceed the currently approved plan, get management approval before proceeding.

Minimize the Development Schedule

The proven methods to minimize the development schedule are straightforward but not always easy: increase the project staff, minimize the amount of rework, and reduce the amount of work. Rarely are there questions about how to increase project staff or reduce the amount of work; the problem is in actually implementing the obvious solutions. Minimizing rework, however, is a quality problem that is more challenging to solve. Here again, there are known and proven methods, and they rest on five principles:

1. It costs more to build and fix a defective product than it would have cost to build it properly the first time.
2. It costs more to fix a defective product after it has been delivered to users than it would have cost to fix it before delivery.
3. It costs more to fix a product in the later testing stages than in the earlier design

and development stages.

4. It costs less to fix product requirements and specification errors in the earlier requirements and specification stages than in the later design and implementation stages.
5. It is least expensive to prevent defects entirely.

A Scalable Process

The fourth requirement for a process to be suitable for large-scale system-development work is that it be scalable. To be scalable, a process must meet the following three criteria:

1. It must use robust and precise methods at all levels, especially at the working systems-engineer and software-development level.
2. Technical and project decisions must be based on and give greatest weight to the knowledge and judgment of the development-level professionals.
3. The process must consistently use data that are derived from accurate, precise, and auditable process and product measurements.

These three points are not generally as widely understood as the cost and schedule problems and call for further elaboration.

Method Scalability--A truly fundamental problem in developing the large-scale systems of the future is that commonly used development methods are nearing their scalability limits and will likely be incapable of handling the much larger challenges of the future. As system scale increases, new methods are generally needed, but the smaller scale methods used in building the supporting system foundations must also be suitable. For software, the principal concern is with the design and quality-management methods commonly used. When developers design and implement the modules and components of massive systems, they typically use the same methods they used with small stand-alone programs. These methods are often little different from the methods they used to write the practice programs they developed when first learning to program. That would be like using the same practices to build the foundation for a 100-story building that you used in building a one-story structure. However, in software development, we do not just scale up by 100 times, we often scale up by 1,000 or 10,000 times, and we typically continue to use the identical development methods, regardless of the job's size.

Management Scalability--Management scalability concerns the estimating, planning, job assignment, tracking, reporting, and control of development work. The current common practice is for managers to make the plans, allocate the work, report progress, and provide project control. However, the suitability of project planning, work allocation, tracking, and control decisions depends on the level of available project information. Furthermore, the level of available project information is inversely proportional to management level. It is therefore clear that this common management-centered planning, allocation, tracking, and control process will not likely work very well when scaled up to support massive systems-development programs.

Measurement Scalability--Why is it that financial accounting methods work for very small organizations and are equally effective for very large corporations? The fundamental reason is that the financial community uses precisely defined methods and auditable data. These methods also assume that the data can be in error, and they include consistent auditing and checking practices to ensure that the data at every organizational level are both accurate and precise. Few of the methods commonly practiced in the development of software-intensive systems use data of any kind, and those that do typically rely on data gathered by other groups, such as accounting, testing, and field support. To have a scalable process, all of the management, technical, and quality practices used in that process must use data that are derived from complete, accurate, precise, and auditable process and product measurements.

Predictable Quality

The fifth and final requirement for a process to be suitable for large-scale work is that it be able to produce quality products predictably. This is one of the greatest challenges

faced by the software community today. The problem is that current practices do not acknowledge that no testing program can identify more than a small percentage of the defects in a large and complex product, and that the larger and more complex the product, the smaller this percentage will be. Further, this means that to get a high-quality product out of testing, one must put a high-quality product into testing. The requirements for a process that will do this are as follows:

- The process must include a family of early defect-prevention and defect-detection activities.
- The process must define and use measures that verify *process* quality during process enactment, and these measures must be applied at every management and development level. This will ensure that all or almost all product defects are found and fixed before testing begins.
- The process must also include measures that verify *product* quality, both before and after testing.
- The process must ensure that quality plans are produced and reviewed regularly, and that deviations from these plans are detected and corrected promptly during process enactment.

Today's commonly used software-development processes do not incorporate quality measurement and analysis. This is a crucial failing since, with even rudimentary measures, the solution to the software-quality problem would have been obvious, and sounder and more efficient software-quality practices would have long since been adopted. The simple fact is that without measurements, no serious quality program can be effective. While developers can make rudimentary quality improvements without measures, to achieve the defect levels required in modern complex systems, we must strive for defect levels of a few parts per million. Such levels are not achievable without complete, consistent, precise, and statistically based quality measurement and analysis.

Conclusions

Development processes must meet the following requirements to successfully produce the large, complex, and critical systems of the future:

- control development cost and schedule predictably
- handle changing needs responsively
- minimize the development schedule
- be scalable
- produce quality products predictably

Finally, the most critical point of all is that a process is of no value if people do not use it. This is the topic for the remainder of this series on large-scale work. For further information about how to improve large-scale systems-development processes, see my SEI technical report on the subject, *Systems of Systems: Scaling Up the Development Process* [Humphrey 06].

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 Dan Burton, Bob Cannon, David Carrington, Marsha Pomeroy Huff, and Bill Nichols.

In Closing, an Invitation to Readers

In this series of columns, I discuss some of the development issues related to large-scale projects. Since this is a complex subject, I cannot hope to be comprehensive, but I do want to address the issues that interest you most. So, if there are aspects of this subject that you think are particularly important and would like to see covered, please drop me a note with your comments, questions, or suggestions. Better yet, include a 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

Reference

[Humphrey 06]
Humphrey, Watts S. Systems of Systems: Scaling Up the Development Process (CMU/SEI-2006-TR-017). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2006.

About the Author

Watts S. Humphreyfounded 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.

Duties, Skills, and Knowledge of Software Architects, The

NEWS AT SEI

Authors

Paul C. Clements

Rick Kazman

Mark H. Klein

This article was originally published in News at SEI on: March 1, 2007

Much research in the software architecture field in recent years has focused on technical aspects of architecting: architectural styles, documentation, analysis, architecture description languages, architectural reverse engineering, and so forth [Bass 03]. But what does it mean for an architect or an architecting organization to be competent? We assert that duties, skills, and knowledge form a triad on which architecture competence rests.

Skills and knowledge support the ability to perform the required duties. An omniscient, infinitely skilled architect is of no use if he or she cannot (for whatever reason) perform the duties required of the position; we might say that person possesses great potential, but we would not say that person is competent.

In particular, we are working toward understanding the following:

- What does it mean to be a competent architect? How can we measure the architecture competence of an individual?
- What does it mean to be an architecturally competent organization? How can we measure the architecture competence of an organization?
- How can an individual increase his or her architecture competence over time? How can an organization increase its overall architecture competence?

These questions define a large research area that is only now beginning to be explored. The goal of competence, of course, is to reliably produce high-quality architectures, but there is much more to it than that. For one thing, architects spend much of their time

doing things other than producing an architecture for the current project. For another, organizations must put in place sound practices and reliable support to assure high-quality architectures in the future; focusing only on the existing architecture(s) will eventually lead to failure.

The desired output of this program of research is to propose a working theory of architecture competence and a framework for discussing and, eventually, measuring competence. This theory may be informal at first, but at the least we expect that it will provide some practical guidance. Perhaps initially the model will consist simply of lists of duties, skills, and knowledge that an individual or an organization must master, but eventually these lists will be replaced with a more detailed understanding of the relationships between what an architect (or architecting organization) does and the effects those things have on the quality of architectures produced both now and in the future.

At the SEI, we are working on creating this theory of competence. Our work began with an attempt to catalog duties, skills, and knowledge. There is no single definitive or authoritative source for the duties, skills, and knowledge required for competence in architecture. There are, however, a number of community resources that we have canvassed to assemble a description of what an architect and an architecting organization must know and do. The sources can be roughly divided into three categories:

1. **Broadcast sources** are sources of information written by self-styled experts aimed at mass anonymous consumption. These sources include
   - *Web sites related to software architecture*. We performed a Web search for sites describing or giving advice on software architecture. Well-known examples include the Bredemeyer Consulting Web site and the Software Engineering Institute's (SEI's) architecture Web site. We took data from the 16 Web sites we found that explicitly mentioned duties, skills, or knowledge.
   - *Blogs and essays related to software architecture*. Rob van Ommering's *Things to Do in Denver If You're an Architect* is a good example of an essay that explicitly discusses architectural duties, skills, and knowledge. In all, we took data from 16 essays.
   - Books on software architecture. By looking at the detailed tables of contents of the Amazon.com best-selling books on software architecture, we can infer what authors are prescribing that architects do, have, and know. The 25 best-selling titles were surveyed; 23 made contributions.
2. **Sources of training and education** tell us what organizations in the business of education or training think that architects need to know and what architects (or aspiring architects) are paying money to learn. These sources include
   - *University courses in software architecture*. A Web search revealed 29 academic courses in software architecture. Course descriptions and syllabi provided lists of duties, skills, and knowledge being taught in these courses.
   - Public non-academic (industrial) courses in software architecture. We gathered data from 22 industrial courses whose descriptions were available online.
   - *Certificate and certification programs for software architects*[1]. Seven programs were identified from which we took data.
3. **Sources related to software architecture careers** tell us what employers are looking for and what architects seeking employment are saying about themselves. This category turned out to be an especially rich source of duties, skills, and knowledge, often explicitly listed in exactly those terms. These sources include
   - *Job descriptions for software architects*. We visited the Web

(< 5 minute) survey.

sites of the top 150 Fortune 500 companies and searched for position descriptions for software architects. We also visited major employment sites. Sixty job descriptions were included in the survey.

- Résumés of software architects seeking employment. We collected about a dozen résumés from employment sites.

Altogether, we gathered information from more than 200 separate sources for our study of architecture competence. The survey resulted in more than 400 duties, skills, and knowledge areas, each of which somebody thinks is important for software architects to master. The collection includes about 200 separately cataloged duties, about 100 skills, and about 100 areas of knowledge. To extract order from the chaos, we performed an affinity diagram exercise to add structure to the duties, skills, and knowledge [Tague 04].

As a result of our survey of publicly available resources, we now have a set of raw data (as described by Clements et al [Clements 07]) that we have distilled into the following broad categories:

Duties:

- architecting
- managing/interacting with life-cycle phases other than architecture
- technology related
- interacting with stakeholders
- management
- organization and business-related issues
- leadership and team building

Skills:

- communication skills
- interpersonal skills
- work skills
- personal skills

Knowledge:

- computer science knowledge
- knowledge of technologies and platforms
- knowledge about organizational context and management

We realized, from our many interactions with practicing architects, that their jobs are far more complex than just making technical decisions, although clearly that remains their most essential single duty. And so we also realized that there was an unfulfilled need for a comprehensive look at what it means for someone to be a competent architect in all dimensions. We envision that this set of duties, skills, and knowledge will be a starting point for an individual wishing to be an architect to determine what he or she should study or practice. Similarly, an organization that wants to nurture architects will benefit from such a categorization, as they will have the beginnings of a roadmap for training and mentoring.

References

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

[Clements 07]
Clements, P.; Kazman, R.; Klein, M.; Devesh, D.; Reddy, S.; & Verma, P. "The Duties, Skills, and Knowledge of Software Architects," *Proceedings of the Sixth Working*

IEEE/IFIP Conference on Software Architecture (WICSA). Mumbai, India, January 6-9, 2007.

[Tague 04]
Tague, N.R. *The Quality Toolbox*, 2nd ed. Milwaukee, WI: ASQ Quality Press, 2004.

[1] A certificate attests to successful completion of a course of study. A certification attests to tested mastery of information or abilities.

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.

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.

Mark Klein is a senior member of the technical staff at the SEI where he is a technical lead in the Architecture Tradeoff Analysis Initiative. He has more than 20 years of experience in research and technology transition on various facets of software engineering and real-time systems. Klein's most recent work focuses on the analysis of software architectures. 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. He has co-authored many papers and is a co-author of the RMA Handbook, *A Practitioner's Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems*.

For more on the duties and skills of software architects, listen to the podcast by Paul Clements.

Other recommended reading

- Architecture capability assessment
- Assessing and Improving Architecture Competence

Share This Page

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

The Mexican TSP Initiative: Positioning the Mexican Software Industry through TSP/PSP

NEWS AT SEI

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

Process Improvement

TSP

This article was originally published in News at SEI on: January 1, 2007

The Software Engineering Institute (SEI), in an alliance with Mexico's leading private university, Instituto Tecnológico de Estudios Superiores de Monterrey (Tec de Monterrey), has engaged in the Mexican TSP Initiative, which had its kickoff in mid-2006 and began launching TSP development teams in December. Carnegie Mellon University President Jared Cohon and Tec de Monterrey Rector Rafael Rangel signed the memorandum of understanding to begin the alliance in March 2006 in Monterrey.

Tec de Monterrey, with support from the Mexican national government and the state governments of Nuevo Leon and Jalisco, is leading this national initiative to position the Mexican software industry as an international competitor. At the heart of the initiative is the proposition that software development can be improved on a national scale by the careful, staged adoption of the SEI's Team Software Process and Personal Software Process (TSP and PSP) methodologies.[1]

TSP and PSP are process technologies that were developed initially at the SEI by SEI Fellow Watts Humphrey and promulgated by an SEI project led by Jim Over. Among the objectives of TSP/PSP are

- reduce time to market
- increase productivity
- improve cost and schedule performance
- improve product quality
- accelerate process improvement
- reduce professional staff shortage

Related Links

News

Heartbleed: Analysis, Thoughts, and Actions

TSP Symposium 2014 Goes Beyond Methodology to Focus on Software Quality

See more related news »

Training

See more related courses »

Events

Team Software Process (TSP) Symposium 2014

Nov 3 - 6

Results with software projects in major software companies that are recognized as global leaders demonstrate that when TSP and PSP are introduced with a competent transition strategy-led by SEI experts and SEI-trained and authorized personnel—the performance of software engineering teams is brought up to world-class levels within the first year—indeed software defects in the field decrease by an order of magnitude. Software development teams are able to forecast delivery dates with little error and can determine at the start of a project the resources that will be needed. Furthermore TSP works well with other improvement methodologies including, among others, Capability Maturity Model[2] Integration (CMMI) and SixSigma.

Positioning the Mexican Software Industry

Leading Mexican software development companies are already successful in the global competition to provide software services. However, the industry is small, with about US$500 million per year accruing from the provision of software-development services to the international community. Successful SEI-led transition of TSP and PSP into select Mexican companies—a demanding first objective—does not reveal the fullness of the Mexican TSP Initiative. Those implementing the initiative have identified TSP as a methodology that will enable the Mexican software industry to achieve targeted growth against significant and established competition. The objective of the initiative is to build sufficient expertise in Mexico to make the country self-sufficient in teaching, applying, and improving TSP.

The Mexican TSP Initiative includes pilot projects at Softtek and at IBM Mexico. Activities are under way in both organizations. Students at Tec de Monterrey have already earned the title of SEI-Certified PSP Developer by passing the SEI's related certification examination. Tec de Monterrey, a university system of more than 100,000 students at 34 campuses throughout Mexico, is also mounting a focused teacher training initiative. The first group of Mexican professors and other training professionals are formally studying PSP from Tec de Monterrey's first SEI-authorized instructor, Rafael Salazar, professor of computer science. Members of this class earned authorization as PSP instructors in January through the follow-up instructor-training class taught by TSP team members Bob Cannon and Jim McHale. Many from this same group are expected to go on to TSP Coach Training, and when successfully observed, they will form the first broad cadre of a more and more ***Mexican*** TSP Initiative.

It is important to the Mexican TSP Initiative that the pilot projects at IBM Mexico and Softtek succeed because the initiative aims to strengthen significantly the productivity, quality, and estimation accuracy of development teams in these Mexican software companies, which will set the standard for the nation. The SEI's TSP team has recognized that a company enjoys its greatest success when the company makes TSP a part of the company *way of doing business*, not only by practicing TSP in software projects, but also through institutionalization by the human resources department in job titles and career paths. Both Softtek and IBM Mexico are following the recommended TSP transition strategy of building internal capability for PSP instruction and TSP coaching. The companies involved in this first phase of the Mexican TSP Initiative are familiar with the SEI. IBM Mexico was the first company in Mexico to earn a CMM Level 5 appraisal (now CMMI Level 5) and Softtek was the first Mexican-owned company to be appraised at CMMI Level 5.

Leaders of the Mexican TSP Initiative envision transitioning TSP to literally thousands of companies—from the largest such as IBM Mexico and Softtek to the very small organizations that employ the majority of Mexican software developers. Clearly the SEI is not able to provide training and coaching on such a grand scale. Not even Tec de Monterrey has the resources to do this. Thus, the next phase of the Mexican TSP Initiative calls for bringing additional Mexican universities and training organizations into the effort. Serendipitously this has already begun. The founders and principal participants of the Mexican organizations Centro de Investigacion en Matematicas A.C. and Quarksoft are authorized PSP instructors and TSP coaches. Their training began during their years at Carnegie Mellon University when they were studying in the Master of Software Engineering program within the School of Computer Science. Because of the independent efforts of these people, TSP is now broadly taught in the states of Zacatecas and Guanajuato.

The Mexican TSP Initiative was precipitated, in part, by the Program for the Development of the Software Industry (PROSOFT). PROSOFT, a Mexican national program under the aegis of the Mexican Ministry of Economy, is dedicated to improving the Mexican economy by improving software development in Mexico. A major interest on the part of the Mexican government is the successful inclusion of small- and medium-sized companies.

Key to the success of TSP is its high-fidelity practice, and key to the high-fidelity practice of TSP is a thorough understanding of PSP. Leaders of the initiative recognize this and have proactively sought ways to ensure and to signal that the Mexican TSP Initiative is high in the quality of PSP instruction and faithful to TSP practice. One way they are doing this is by embracing SEI certifications. They have already begun having students sit for the SEI-Certified PSP Developer examination. Plans call for broad use of the certified developer program. They are looking forward to the upcoming TSP coach certification and are exploring other methods to ensure excellence, such as using feedback to control the quality of PSP instruction.

The Mexican TSP Initiative will provide mentoring for its instructors by using the SEI Blended Learning offering of PSP for Engineers. SEI Blended Learning incorporates an online training environment and remote, SEI-approved mentors to deliver SEI courses. Through this model, best practices will be shared, and the quality of work by students and instructors will be monitored as SEI-based mentors review the audit trail of educational artifacts—providing timely feedback to instructors on how to best guide their students in the acquisition of PSP skills. Although final pricing remains to be set, it is already clear that the SEI can provide mentoring through the SEI Blended Learning courses at a fraction of the cost of teaching face to face. Similar leveraging of SEI resources in the arena of coaching is not yet in place but is a topic of investigation.

The Mexican TSP Initiative has impressive goals. It is an ambitious project that has captured the imaginations of many. On the TSP team, Over and Humphrey are enthusiastic supporters of the initiative, and both have traveled to Mexico to speak and teach classes. At the official kickoff of the Mexican TSP Initiative, to the enthusiastic approval of those in attendance, Humphrey articulated the team's shared objective, "In five years I want the world to be asking, '*How did Mexico do it?*'"

[1] Team Software Process, TSP, Personal Software Process, and PSP are service marks of Carnegie Mellon University.

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

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

SEI Reaches Across Domains to Aid Health Care Industry

NEWS AT SEI

This article was originally published in News at SEI on: January 1, 2007

U.S. military agencies depend on accurate information to determine quickly where, when, and how to respond to threats. A complex array of data, fed from a variety of sources, must be automatically shared, analyzed, and filtered. Only then can the military receive the precise details needed to make rapid, informed decisions.

The U.S. Army is refining its net-centric capabilities, an effort that has been supported by the SEI. Since 1985, the SEI's research has refined and improved many of the software-intensive systems that support an extensive U.S. Department of Defense (DoD) IT infrastructure, which must allow underlying systems and applications to work together flawlessly.

The SEI works with the DoD and others to improve the systems that support everything from logistics and acquisitions to day-to-day business operations and net-centric battlefield management.

Now the SEI is transferring its battle-hardened expertise to health care.

Similar Challenges

"We were struck by the interesting similarities in some health-care situations and some military situations," says Suzanne Garcia, a senior member of the SEI's technical staff. "Logistics management offers one example. Getting supplies, parts, and people to the right place at the right time within a military context correlates to the logistics challenges that large health systems with multiple facilities, departments, and medical disciplines also face."

Another similarity relates directly to battlefield management.

"Just as a war fighter bases decisions on complex data that are fed from multiple sources, an emergency-room or intensive-care clinician also makes rapid patient-care decisions based on complex data that are fed from multiple sources," says Garcia.

The DoD saw improvement when it began to tear down communication barriers among its IT systems; likewise, improved communication is a key challenge for the

health care industry. How can the IT systems that support doctors, nurses, emergency-room clinicians, laboratories, insurance providers, and patients provide real-time access to accurate and complete patient data?

That's where the SEI enters the picture.

Matching Health-Care Challenges with SEI Capabilities

To date, the SEI has conducted assessments and research for the U.S. Department of Veterans Affairs and the University of Pittsburgh Medical Center. The SEI recently chaired an International Process Research Consortium (IPRC) workshop that focused on process challenges in health care. "In the course of these activities, leaders in health care and health IT are identifying challenges that can be addressed with SEI capability," says Eileen Forrester, senior member of the technical staff at the SEI and co-chair of the IPRC. "Obviously, the SEI capability in process improvement is relevant, but there are other challenges that match our capabilities too." Those challenges include the following:

Interoperability—The power of the Internet has dramatically improved interoperability across the military. The SEI has helped the DoD and other organizations use the Internet, among other tools, to rapidly transfer and share information among IT systems that previously had not been able to communicate.

"The question is, how do we apply these interoperability principles to health care?" says Forrester. "How do we ensure that an intensive-care clinician is able to access the same test results during a weekend emergency as that patient's doctor would during a standard office visit?"

To foster interoperability in IT systems, the SEI established the Integration of Software-Intensive Systems (ISIS) initiative. The SEI has found that achieving interoperability is much more than an engineering problem. ISIS is pursuing several projects that deal with the multiple dimensions of interoperability challenges. A common issue they all face is stakeholder management and coordination.

"In health care, all the stakeholders of an IT system, especially the end users, need to have a strong voice early in the development process, and throughout that process, to help identify and address interoperability issues," says Garcia. "Likewise, clinical users of health-care systems must be actively involved with the vendors and IT personnel who are making decisions about their systems. This helps ensure that the systems are not overly constrained—that interoperability will be more easily accomplished as new systems are added in the future."

Security—In the military, preventing information from getting into the wrong hands could be a matter of life or death. As health-care providers begin to make medical records widely available over the Internet, one challengeis to protect data from malicious users while allowing legitimate users quick and easy access.

"Standards such as the HIPAA Privacy Rule mandate that entities receiving health care information from consumers adequately protect the data," says Garcia. The HIPAA Privacy Rule is a regulation of the federal Health Insurance Portability and Accountability Act (HIPAA). "The SEI has developed techniques for assessing security risks for the DoD that are helping to identify and mitigate weaknesses within the health-care domain. The early pilots of security risk assessment were performed in the DoD health system environment."

Adoption—One of the larger challenges with technology in any domain is not the technology itself, but getting stakeholders to apply it and use it. Working with the DoD and its suppliers, the SEI developed a set of techniques that support comprehensive organizational change when adopting new processes and technologies.

"The SEI has a strong history and experience in understanding the adoption of new practices, such as using a new logistics-management tool or using a new protocol for making decisions of a particular type," says Garcia. "Our adoption techniques apply equally well to clinicians adopting a new electronic medical-record system or a logistics specialist adopting a new commercial scheduling package for vehicle-maintenance management."

Adoption challenges are similar among domains, and the SEI has identified some of the acute challenges in health care. "For example, in every domain, our adoption processes must account for the fact staff members have competing commitments for their time," says Forrester. "In health care, it's hard for doctors, nurses, or lab technicians to justify setting aside the time to learn a new software application when patient-care activities are waiting for their attention."

When possible, the SEI looks for opportunities to mitigate such a challenge in ways that are specific to that context. One of the challenges in electronic medical record systems is how to represent the tremendous amount of information that is present in a patient's chart. Some users want to see the electronic format mimic the paper chart they are already familiar with—but this may introduce problems and miss opportunities. So should the electronic system try to mimic a paper chart to win over some users, or should it try to find ways to represent the information that take advantage of computer-user interface strengths that go beyond the paper-chart approach? "We've found that, predictably, some clinicians adapt most easily to an electronic record that mimics a paper chart, and others adapt fairly easily to novel information designs," says Garcia. "One of our adoption techniques can help organizations determine which approach is likely to be most successful with their clinicians."

Shared Solutions

During the past 20 years, the SEI has served as a neutral advocate of the best software engineering practices—an objective voice that has helped the DoD establish parameters and standards that get a wide variety of stakeholders to improve their results in acquiring, developing, and operating IT systems. The SEI and DoD are currently working for dramatic improvement in interoperability across diverse IT systems in DoD and industry.

Today, health care faces similar challenges—it also requires a neutral advocate to get all the stakeholders in concert. "The process of incorporating and adopting new ideas around health-care challenges will not only help deliver robust practices to the health care community," says Garcia, "but it will help us further enrich our understanding of a broad range of issues that we can then transfer back to the DoD and the other domains we support."

The SEI Chairs IPRC Workshop on Health Care

The International Process Research Consortium (IPRC) is dedicated to projecting future needs and trends in process research over the longer term (five to 10 years in the future). IPRC members include internationally recognized experts from all over the world as well as organizations in such divergent domains as manufacturing, consulting, research and health care. During the first week of November 2006, the SEI led an IPRC workshop among health care stakeholders to address the unique process needs and challenges faced by the health care domain.

About the Author

Tom Purcell is a freelance writer and nationally syndicated columnist.

Protecting Against Insider Threat

NEWS AT SEI

Authors

Dawn Cappelli (CERT)

Andrew P. Moore

Timothy J. Shimeall

This article was originally published in News at SEI on: February 1, 2007

Are Insiders Really a Threat?

The threat of attack from insiders is real and substantial. The 2006 E-Crime Watch Survey conducted by the United States Secret Service (USSS), the SEI CERT Program, and *CSO Magazine*, found that in cases where respondents could identify the perpetrator of an electronic crime, 32% were committed by insiders. The impact from insider attacks can be devastating. One complex case of financial fraud committed by an insider in a financial institution resulted in losses of almost $700 million. Another case involving a logic bomb written by a technical employee working for a defense contractor resulted in $10 million in losses and the layoff of 80 employees.

Over the past several years, CERT has been conducting a variety of research projects on insider threat. One of the conclusions reached is that insider attacks have occurred across all organizational sectors, often causing significant damage to the affected organizations. These acts have ranged from low-tech attacks, such as fraud or theft of proprietary information, to technically sophisticated crimes that sabotage the organization's data, systems, or network. Damages are not only financial; widespread public reporting of the event can also severely damage the organization's reputation.

Insiders have a significant advantage over others who might want to harm an organization. Insiders can bypass physical and technical security measures designed to prevent unauthorized access. Mechanisms such as firewalls, intrusion-detection systems, and electronic building-access systems are implemented primarily to defend against external threats. However, not only are insiders aware of the policies, procedures, and technology used in their organizations, but they are often also aware of their vulnerabilities, such as loosely enforced policies and procedures or exploitable

technical flaws in networks or systems.

Partnering with the USSS, CERT has been conducting the *Insider Threat Study*, gathering extensive insider threat data from more than 150 case files of crimes involving most of the nation's critical infrastructure sectors. To date, researchers have published two reports documenting the results of the study: Insider Threat Study: Illicit Cyber Activity in the Banking and Finance Sector and Insider Threat Study: Computer System Sabotage in Critical Infrastructure Sectors. This study shows that use of widely accepted best practices for information security could have prevented many insider attacks or detected them earlier. Rather than requiring new practices or technologies for prevention of insider threats, the research instead identifies existing best practices critical to the mitigation of the risks from malicious insiders.

Who Is The Suspicious Insider?

Disgruntled technical staff members, both before and after termination, must be recognized as potential threats for insider IT sabotage. Data pertaining to fraud and information theft suggest that organizations must exercise some degree of caution with *all* employees. Current employees in practically any position have used legitimate system access to commit these types of crimes. (Of special note is that almost half of the employees who stole information while still employed had already accepted other job offers.) Unfortunately, there is no profile of an insider who poses a threat to an organization; the threat can be recognized based only on a combination of patterns of behavior and online activity.

Can Insiders Be Stopped?

Insiders can be stopped, but stopping them is complex. Insider attacks can be prevented only through a layered defense strategy consisting of policies, procedures, and technical controls. Therefore, management must pay close attention to many aspects of an organization, including its business policies and procedures, organizational culture, and technical environment. Managers must look beyond information technology to the organization's overall business processes and the interplay between those processes and the technologies used.

Too often organizations allow the quality of their practices to erode as no malicious activity is detected over time. One of the vulnerabilities posed by insiders is their knowledge of exactly this: the quality of their organization's defenses. Based on our research to date, the practices outlined below are the most important for mitigating insider threats.

Practices for Preventing Insider Attacks

The following 13 practices for preventing insider attacks will provide an organization with defensive measures that could prevent or facilitate early detection of many of the insider attacks other organizations have experienced. This is an overview of the best practices covered in the "Common Sense Guide to Prevention and Detection of Insider Threats, 1st Edition; see the complete document for more details.

Practice 1: *Institute periodic enterprise-wide risk assessments.*

It is difficult for an organization to determine the proper balance between trusting its employees, providing them access to achieve the organization's mission, and protecting itself from those same employees. Access combined with knowledge of the organization's vulnerabilities in both technology and business processes gives insiders the ability to carry out malicious activity against their employers. An organization must protect itself from both insiders and outsiders using risk-management principles. The organization must take an enterprise-wide view of information security, first determining its critical assets, then defining a risk-management strategy for protecting those assets from both insiders and outsiders.

Practice 2: *Institute periodic security awareness training for all employees.*

A culture of security awareness must be instilled in the organization so that all employees understand the need for policies, procedures, and technical controls. The first line of defense from insider threats is the employees themselves. All employees in an organization must understand that security policies and procedures exist, that there is a good reason that they exist, that they must be enforced, and that there can be

serious consequences for infractions. Each employee must be aware of the organization's security policies and the process for reporting policy violations.

Practice 3: *Enforce separation of duties and least privilege.*

If all employees are adequately trained in security awareness, and responsibility for critical functions is divided among employees, the possibility that one individual could commit fraud or sabotage without the cooperation of another individual within the organization is limited. Effective separation of duties requires the implementation of *least privilege*, that is, authorizing people only for the resources they need to do their jobs.

Practice 4: Implement strict password and account-management policies and practices.
No matter how vigilant employees are in trying to prevent insider attacks, if the organization's computer accounts can be compromised, insiders have an opportunity to circumvent both manual and automated mechanisms in place to prevent insider attacks.

Practice 5: *Log, monitor, and audit employee online actions.*

If account and password policies and procedures are enforced, an organization can associate online actions with the employee who performed them. Logging, periodic monitoring, and auditing provide an organization the opportunity to discover and investigate suspicious insider actions before more serious consequences ensue.

Practice 6: *Use extra caution with system administrators and privileged users.*

Typically, logging and monitoring is performed by a combination of system administrators and privileged users. Therefore, additional vigilance must be applied to those users.

Practice 7: *Actively defend against malicious code.*

System administrators or privileged users can deploy logic bombs or install other malicious code on the system or network. These types of attacks are stealthy and therefore difficult to detect in advance, but practices can be implemented for early detection.

Practice 8: *Use layered defense against remote attacks.*

If employees are trained and vigilant, accounts are protected from compromise, and employees know that their actions are being logged and monitored, disgruntled insiders will hesitate to attack systems or networks at work. Insiders tend to feel more confident and less inhibited when they have little fear of scrutiny by coworkers; therefore, remote-access policies and procedures must be designed and implemented very carefully.

Practice 9: *Monitor and respond to suspicious or disruptive behavior.*

In addition to monitoring online actions, organizations should closely monitor other suspicious or disruptive behavior by employees in the workplace. Policies and procedures should be in place for employees to report such behavior when they observe it in coworkers, with required follow-up by management.

Practice 10: *Deactivate computer access following termination.*

When an employee terminates employment, whether the circumstances were favorable or not, it is important that the organization have in place a rigorous termination procedure that disables all of the employee's access points to the organization's physical locations, networks, systems, applications, and data.

Practice 11: *Collect and save data for use in investigations.*

Should an insider attack, it is important that the organization have evidence to identify the insider and follow up appropriately.

Practice 12: *Implement secure backup and recovery processes.*

Despite all of the precautions implemented by an organization, it is still possible that an insider will attack. Therefore, it is important that organizations prepare for that possibility by implementing secure backup and recovery processes that are tested

periodically.

Practice 13: *Clearly document insider threat controls.*

As an organization acts to mitigate insider threat, clear documentation will help to ensure fewer gaps for attack, better understanding by employees, and fewer misconceptions that the organization is acting in a discriminatory manner.

Additional Resources

Cappelli, Dawn; Moore, Andrew; & Shimeall, Timothy. *Common Sense Guide to Prevention and Detection of Insider Threats, 1st Edition*. Pittsburgh, PA: Carnegie Mellon University CyLab, 2005.

"CERT Execs on the 2006 E-Crime Watch Survey," *CSO* (podcast) (September 2006).

CERT Insider Threat Research (2007).

Rasmussen, Gideon. *Insider Risk Management Guide* (August 30, 2006).

Insider Threat.

About the Authors

Dawn Cappelli is a senior member of the technical staff with the CERT Program at the SEI. She is technical lead of CERT's insider threat research, including the *Insider Threat Study* conducted jointly by the U.S. Secret Service and CERT. Other current work includes modeling and simulation projects for risk analysis and communication of impacts of policy decisions, technical security measures, psychological issues, and organizational culture on insider threats. Cappelli is also an adjunct professor in the Carnegie Mellon H. John Heinz School of Public Policy and Management.

Cappelli has been with Carnegie Mellon since 1988. Before joining CERT in 2001, she was director of engineering for the Information Technology Development Center of the Carnegie Mellon Research Institute, led special projects for the university's Computing Services, and worked on projects for the Software Engineering Institute's Information Technology team.

Andrew Moore is a senior member of the technical staff with CERT. He is also a research scientist at Carnegie Mellon CyLab. Moore explores ways to improve the security, survivability, and resiliency of enterprise systems through attack and defense modeling, incident processing and analysis, and architecture engineering and analysis. Before joining the SEI in 2000, he worked for the Naval Research Laboratory investigating high-assurance system-development methods for the Navy. He has 20 years' experience developing and applying mission-critical system-analysis methods and tools, leading to the transfer of critical technology to both industry and the military.

Timothy J. Shimeall is a senior member of the technical staff with the CERT. His research interests are include information survivability, network situational awareness, and analysis of security incident behavior. He received his PhD in information and computer science from the University of California, Irvine, in 1989. Before joining the SEI, he served as an associate professor at the Naval Postgraduate School in Monterey, California.

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

CMMI: Product Suite Expansion

NEWS AT SEI

Author

Mike Phillips

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

Process Improvement

CMMI

This article was originally published in News at SEI on: February 1, 2007

My previous column summarized the familiar elements of the CMMI Product Suite and the beginnings of the transition to the newest version, V1.2. This column introduces a new component of the CMMI Product Suite: *Understanding and Leveraging a Supplier's CMMI Efforts: A Guidebook for Acquirers*, otherwise known as the *Acquisition Guidebook*. This guidebook, which will be available online by the end of March, links the existing product suite to the needs of government acquisition organizations in selecting qualified suppliers.

CMMs and Government Acquisition

From the inception of the Capability Maturity Model for Software (SW-CMM) in the late 1980s, government officials have sought ways to gain confidence in the capability of the suppliers of software-intensive systems to deliver products on time and with high quality. The SW-CMM was used from the start as a way to help assure that success. A tenet of this focus is that the quality of a product (or software-intensive system) is directly related to the processes used in its development, including the engineering, project-management, and support processes. The CMMI team, which consists of representatives from government, industry, and the SEI, has increased the scope of processes covered beyond that covered by the SW-CMM. Now more information is available to acquirers seeking to evaluate the growing number of suppliers that use CMMI for process improvement.

Myths of What Levels Can Indicate

In an earlier column, I discussed some of the myths about CMMI appraisals. A number

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

of these myths relate to associating a capability or maturity level with an assurance of actual performance. Many Department of Defense and other government systems, however, are developed to include capabilities that have not existed before. Although we believe that organizations with more disciplined processes offer fewer development risks, risks do remain. Another myth is that expectations based on maturity or capability levels can be extrapolated to cover other team members, such as sub-contractors; other company elements; or the acquisition organization.

In the past, acquisition organizations had only a maturity-level or capability-level profile as an indicator of the value of a potential supplier. The *Acquisition Guidebook* provides information on how to interpret maturity levels and capability levels in a way that accurately reflects the strengths and weaknesses of potential suppliers, including what parts of the organization were appraised and what parts of the CMMI model were included in the appraisal, among other things.

Development of the Guidebook

The idea to develop the *Acquisition Guidebook* was based on an event sponsored by the National Defense Industrial Association (NDIA). Workshop participants recognized the importance of providing a sensible guidebook for the acquirer, and the CMMI Steering Group chartered a team to identify the best ways to use CMMI for Development V1.2 (CMMI-DEV) as an acquisition tool. Our DoD sponsor helped shape requirements for the guidebook, including a limit on its size. It was not to grow too large, but it did have to capture the better ways to interpret existing appraisal results to help select a supplier. It also had to help the acquirer consider strategies for future reviews of process improvement with the chosen supplier. As with all of the other elements of the CMMI Product Suite, an integrated team from government, industry, and the SEI created the initial document. A smaller government team then revised the draft to focus it to the essential elements that would be most appropriate for government use. The guidebook presumes that users have limited knowledge of CMMI but that potential suppliers are using CMMI to improve their development processes. This approach to the guidebook helped the CMMI Steering Group to formulate the title: *Understanding and Leveraging a Supplier's CMMI Efforts: A Guidebook for Acquirers.*

The Contents of the Guidebook

The guidebook defines a path for use of CMMI elements by the acquirer in its four chapters:

The first chapter covers the basics of CMMI, including a brief review of the elements of the model, how capability levels and maturity levels complement each other, and how various appraisals can provide insight into the development capabilities of potential suppliers.

The second chapter helps the acquirer relate the model's process areas to the critical elements of the acquisition program. Identifying these relationships enables the acquirer to evaluate potential suppliers using the process areas critical to that program.

The third chapter focuses on ways that the acquirer can leverage the existing process capabilities of the supplier to the advantage of the acquisition program.

The fourth chapter highlights the value of process reviews during the supplier's development lifecycle to monitor and control the supplier's activities and ability to meet the terms of the supplier agreement.

It wasn't clear at first how best to provide additional information for each of the various strategies. In the end, the authors provided a set of appendices to cover the needed details so that acquirers would have guidance on how to implement the approaches. The appendices provide questionnaires and a mapping to the *Defense Acquisition Guidebook*, as well as information on what to do, when to do it, and how to do it for each approach offered.

Summary

The guidebook represents a crucial link, captured in a single document, between the acquirer and the supplier. This document addresses how claims of CMMI Levels may be misleading and encourages a continuous approach to process improvement. A risk

perspective is implied, which focuses acquirers on taking advantage of process improvement in areas most needed for their programs. Acquirers who see the value of these methods may also appreciate and use the upcoming CMMI for Acquisition (CMMI-ACQ) constellation that is currently under development.

My next column will discuss the efforts we are undertaking to continuously improve V1.2 appraisals. While much has been done in the current product-suite release, our approach to the certification of SCAMPI Lead Appraisers will leverage current capabilities while building the professionalism of these important SEI Partners.

About the Author

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.

Large-Scale Work – Part VII: Process Discipline

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: February 1, 2007

This is the seventh and final column in a series on large-scale development work. Part I introduced the structural problems of the massive organizations that typically do large-scale work. Part II addressed the management decision process, why this process is critical for large-scale work, and how to fix it if it is broken. Part III discussed how to get things done in a bureaucratic organization, and Part IV addressed the need for teams to take charge of their own work. Part V then described self-directed teams and how their planning and self-management skills help them to consistently produce superior results. Part VI discussed the requirements a process must satisfy to consistently and predictably produce high-quality large-scale systems. This column discusses how to get the developers to follow their selected processes in doing their development work. As we will see, this challenge is not trivial, but it is also not hopeless.

Process Requirements

In Part VI, we discussed the requirements for a process that will predictably produce high-quality large-scale systems. In summary, the five basic requirements are to

1. control development costs and schedules predictably
2. handle changing project needs responsively
3. minimize the development schedule
4. be scalable
5. produce quality products consistently

While the actions required to accomplish all of this are not particularly complex, they do require that all of the systems, software, and hardware developers and their teams consistently follow certain essential practices. The five required practices are to

1. estimate, plan, and track personal and team work

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

2. estimate the impact of every change and negotiate needed schedule or resource adjustments before implementing the change
3. consistently use the team's or project's selected methods for every step of the work
4. plan, measure, and manage the quality of every part of the process
5. take immediate corrective action whenever the quality of any product or product element fails to meet team or project standards

For people to work in this way, they must have the proper skills, be highly motivated, and have the leadership and support to consistently do superior work.

Skill Requirements

The skills required fall into three categories: technical, project, and quality.

Technical Skills—Although the required technical skills are by far the most complex, difficult, and time-consuming to learn, they are not usually the source of most project problems. Technical topics are the principal focus of current academic curricula, and these are the topics the developers find most interesting. Yet even though insufficient technical skills are not usually the cause of project problems, when such skills are insufficient, projects invariably have problems. In short, suitable technical skills are essential, and without them, even the most motivated and best-led and supported teams will not likely succeed, at least not on predictable schedules.

Project Skills—Project-related skills principally concern estimating, planning, tracking, and configuration management. While these skills are relatively easy to learn and can be taught in only a few days, without them, developers and their teams cannot do predictable work. The main reason that such skills are not required on most projects is lack of understanding by the developers and their management. Technical professionals are not taught self-management skills during their educations. While this self-management problem has been particularly serious for software, it is also a problem for every development discipline.

This is an unfortunate failure because truly superior work is invariably done by people who manage themselves. Unfortunately, if technical professionals do not know how to manage themselves, their managers have to manage them. Then the managers make the plans, define the commitments, allocate the work assignments, and monitor work status and performance. If this does not sound like a fun way to work, that is because it isn't. The only way out of this trap is for developers to start planning and managing their own work.

Quality Skills—While quality problems are not new to the systems-development business, the character of these problems has changed. Early systems had thousands of largely identical parts, and all of these parts were interconnected by wires. The quality challenge at the time was twofold: to find parts of high enough quality and with sufficiently long expected lifetimes to get the systems to work. Then, the priority was to ensure the quality of all the connections among these parts. A useful early test was to literally shake these systems and, if there were intermittent errors, look for and fix the bad connections.

With integrated circuits, these early quality problems were largely solved. Now, the problem is with the multiplicity of part types. We used to have hundreds to thousands of copies of a relatively few part types, but now we also have thousands of different designs. Of course, with thousands of anything, you potentially have quality problems. These problems were first apparent with software, where we never had standardized parts and every module was unique. Large software-intensive systems have always had thousands of parts, and with really large systems, we now even have millions of different designs that all must work correctly.

With software, quality problems have been serious, but they have always been simpler and easier to fix than hardware quality problems. The reason is that, because the costs and time delays associated with fixing a defective software component were much less severe than with fixing defective hardware, it has been practical to fix software components during testing. With hardware, it is impossible to fix defective chips

during testing. As a result, hardware quality has traditionally been viewed as a manufacturing problem and not something that the developers needed to worry about.

With embedded software and with software logic now widely used in hardware design, software quality problems are becoming pervasive in the hardware world. They are also becoming as time-consuming and expensive to fix as traditional hardware problems. In fact, nearly a decade ago, an auto company executive told me that the company had started to make hardware changes to avoid changing the software. So we must all learn to adopt sound quality-management skills. The hardware designers must learn from the manufacturing community, and the software engineers must participate in this learning process.

The skills required for quality management are relatively simple, but they are not easy. All that is required is to use a precisely defined personal process, to track and record every defect, and to use these defect data to modify the process both to prevent defects and to find and fix almost all of the defects before the start of testing. The manufacturing community has defined and refined the required practices. Now the development community must begin to adopt these practices. While this is partly a training problem, the training is relatively simple and can be completed in a few weeks. The most difficult issues are motivation, leadership, and support.

Motivational Requirements

There is only one motivational requirement: motivating the systems, hardware, and software developers to adopt and consistently use sound quality methods. To do this, these developers must be trained in quality-management skills, and they must work on teams in which all the members follow these same practices. However, because gathering and using data are important parts of quality management, and since it is essential that these data be accurate and complete, developers must be interested in their data and motivated to gather and use these data to manage their personal work. If they are not, the data will almost certainly be incomplete and imprecise. Finally, and most important, the entire management team must both motivate the developers to follow their team-defined quality practices and refrain from using any of the resulting data in any way that threatens the team members. The need is for the kind of trusting environment that makes creative work possible. While a trusting environment is important for all engineering programs, it is essential for large-scale work. Without trust, there is only limited communication, and without communication, large programs cannot be managed.

Leadership Requirements

Leadership is the key, but leadership is different from managing. Napoleon, when asked how he made his army cross the Alps into Italy, said: “One does not make a French army cross the Alps; one *leads* it across” [Humphrey 1997]. Leadership requires vision, an ability to define compelling goals, and the foresight to adopt improved practices and methods before they are widely adopted and before the need to use them has become obvious. Leadership involves setting goals, defining directions, and exciting the troops.

In leading large-scale operations, all of the traditional leadership practices are essential, but they are not sufficient. With large engineering programs, the leadership challenge is to build a cooperative management culture that enables rational fact-based decision-making and minimizes political infighting. In one example, the manager of one part of a large program encountered an unanticipated problem and needed additional resources. When he explained this to the leadership team, we knew that we could not expect help from anyone else; we would have to fix the problem ourselves. The managers then asked me to leave so they could try to work out a solution among themselves. When I returned, they had solved the problem. Each manager had looked in his own group to identify any lower-priority work and had either cut or delayed it. With the right kind of culture, managers know that they will get help when they need it and are generally willing to help others when they can.

Leadership is about more than just setting direction, taking risks, and motivating the troops. It involves building a trusting and cooperative culture. It must also recognize and reward superior work. This requires that the developers be properly trained, encouraged to manage their own personal work, have the right kind of support, and be

properly recognized and rewarded. This, of course, means that the leaders must know what superior work looks like, look for and identify such work, and ensure that it is properly rewarded.

Support Requirements

The final need is for support. In addition to the support provided by proper training and effective leadership, there is the need for coaching support. In sports and the performing arts, we all recognize the need for coaches. In fact, when a team is doing badly, it is always the coach who gets blamed. The coach, conductor, or director is the one who guides, motivates, and challenges the team members. However, we in the development community have been slow to recognize the need for coaching support.

Regardless of the field, whenever it is important for people to do consistently high-quality work, teams need coaching support. It is hard to do complex and creative work, and it is doubly hard to do it with extreme care and precision. If no one notices or cares about what we do or how we do it, it is almost impossible to maintain a high level of personal performance. That is when coaching is truly essential: to provide the support, encouragement, and guidance required to maintain motivation and dedication. Without motivation and dedication, there cannot be superior performance.

Conclusions

This concludes this series of columns on large-scale work. Large-scale engineering work involves many challenges. It requires properly designed and structured organizations, sound and minimally bureaucratic decision-making processes at every level, balanced and participative management systems, well-designed and structured development processes, and the leadership and coaching support required for consistently superior work. The methods and practices required to do all of this are well known and available. The principal challenge is to do it.

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 Alan Willett.

In Closing, an Invitation to Readers

In this particular series of columns, I have discussed some of the development issues related to large-scale projects. As I now consider the topics to address in subsequent columns, I would appreciate any of your comments and suggestions. If there are topics that you feel are particularly important and would like to see covered, please drop me a note with your comments, questions, or suggestions. Better yet, include a story or brief anecdote to illustrate 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

References

[Humphrey 1997]
Humphrey, W. S. *Managing Technical People—Innovation, Teamwork, and the Software Process*. Reading, MA: Addison-Wesley, 1997.

[Humphrey 2006]
Humphrey, W. S. *TSP: Leading a Development Team*, Reading, MA: Addison-Wesley, 2006.

About the Author

Watts S. Humphreyfounded 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.

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

SEI Publishes Framework for Software Process Research

NEWS AT SEI

Author

Matt B. Weissberg

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, 2007

Imagine a future in which our already highly integrated world becomes even more interdisciplinary. International investment heavily focuses on areas such as DNA-neural network integration, self-aware medical devices in human systems, and zero-gravity cellular computing environments. How will processes be defined to accommodate the multiple contexts of these varied disciplines?

Now imagine another possible future state, one in which the world faces a global pandemic. As a new disease threatens the majority of the world's population, the Internet becomes the main tool for people to collaborate on improvised survival techniques, conduct commerce, and attend school and church. What processes would keep the Internet sufficiently robust to play this new, expanded role?

In an effort that began in August 2004, the SEI's International Process Research Consortium (IPRC) brought together 27 leaders from academia and industry to study the process-research implications of possible future scenarios such as these, as well as to consider the directions in which existing research is currently pushing the process community. The result of this endeavor is the new SEI publication *A Process Research Framework* (ISBN-13: 978-0-9786956-1-3), which was made available in February.

This new book serves as a guide that strategically formulates questions to help direct future research efforts for the process community. "This SEI initiative drew researchers from around the globe and distilled their combined experience and observations into a cohesive research framework," says George Wilkie, director of the Centre for Software Process Technologies at the University of Ulster, United Kingdom,

Related Links

Training

Software Product Lines - eLearning

Software Product Lines

See more related courses »

and the architect of the framework.

Providing Strategic Guidance for Future Research

Through a collection of research themes and questions, *A Process Research Framework* provides guidance for industry, researchers, and funding agencies in determining the most pressing questions to pose in their research efforts. These programs of investigation can then be more effectively formulated to address strategically important challenges.

Wilkie notes that readers of the framework will take away "an awareness of the major driving forces that will shape the way technological products are engineered in the future. The IPRC has created broad research themes that provide readers with a means of structuring their thinking."

Each research theme embodies a common or recurring topic that appears or is implicit through several of the plausible future scenarios that the IPRC developed. The themes are oriented to enable researchers to close the gap between process capabilities of today and the likely business and technical process needs of tomorrow.

"To our surprise, the four main research themes that evolved were quite familiar," says Eileen Forrester, co-chair of the IPRC and editor of the framework. "Persistent concerns we have today—for example, how to deploy processes effectively or ensure that they produce the results we want—are still relevant in the context of future scenarios.

"Consider how urgent it will be to specify processes that reliably achieve the product qualities we need when facing a scenario such as a global pandemic. Can we ensure that the processes used to build the next version of the Internet will produce something that can function under the stringent conditions we would face during a pandemic? What research questions must we work to answer *now* so that we are ready?"

In addition to the four research themes, the IPRC members consider the influence of emerging trends and technologies on process issues. This assessment is composed of three factors: continuous requirements evolution, incomplete knowledge, and heterogeneous component-based systems integration.

Inside the Research Themes

A Process Research Framework does not provide the answers to solve the future challenges for processes. Instead, the book serves a tool intended to better focus the efforts of the entire process research community in preparing for possible future environments.

Research questions form the base foundation of the framework. The questions are grouped to form cohesive inquiries into specific research topics. Depending upon the challenges that individual framework users are attempting to address, the answers to some of these questions for a specific course of research may be relatively straightforward. For other questions, the answers may require significant programs of research.

In all, the research framework consists of a dozen possible future scenarios, four themes, a consideration of emerging trends, 20 topic-specific research nodes, and more than 230 research questions.

Raising Awareness of Process Issues

"It's all about the evolution of a discipline," emphasizes Wilkie, who was recently recognized as the British Computer Society's IT Professional of 2006. "In the early days of software engineering, there was a tendency to jump straight into the coding with little or no regard given to the design phase."

"We have evolved beyond this mentality but still have a long way to go before full development processes are given the attention they deserve. *A Process Research Framework* makes a key contribution to understanding why process is important and why its importance will only increase as time goes by," Wilkie says.

"The SEI and its partners have been successful in leading the current process-improvement revolution in software, but we can't stop there," says Forrester. "It is also necessary for us to consider what's next for process research. What research issues need attention now if we are to be ready for the world we will live and work in a decade from today?"

Next Steps for the Framework

The current edition of the framework is intended to act as a catalyst for communitywide discussions that might further enhance and expand the research scheme for future versions.

Plans for the next two years include gradually converting the book into an online, interactive site. The larger process community will be able to use the site to contribute to the framework and, in so doing, will be able influence the strategic direction of the field.

Forrester says, "We are eager to get the larger process community participating—we hope they will add content and context, and engage in a dialogue with us on what they think are the highest priority challenges as we prepare for the future. We are encouraged by the response already. Multiple organizations have notified us that they will be using the framework as an input to their strategic planning or suggested research programs in which they would like to participate."

Forrester and Wilkie will be discussing the frameworkat the SEPG Conference in Austin, Texas, on Thursday, March 29, from 4:20 to 5 p.m. The first 50 attendees at the presentation, titled "Developing a Process Research Framework," will receive free copies of the book.

A Process Research Framework is sponsored by BAE Systems, Robert Bosch GmbH, Lockheed Martin Corporation Integrated Systems & Solutions, the SEI, Tata Consultancy Service (USA), and the University of Pittsburgh Medical Center.

For more information regarding *A Process Research Framework*, see www.sei.cmu.edu/iprc/roadmap.html. To purchase a copy of the book, contact us using the link in the *For More Information* box below.

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

Being Your Own Boss—Part I: The Ideal Job

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, 2007

This is the first of a series of columns on our work as developers and how we can have truly satisfying jobs. This first column in the series discusses ideal jobs, what developers like about their work, and what they find annoying and unpleasant. It then describes how the developers' and managers' views on project success differ and what this means for both developers and their organizations. In subsequent columns, I will discuss how to turn almost any job into an ideal job, some of the issues you will face in trying to do so, and the level of support you will need from your managers and peers to be successful. While there are a few basic conditions that must be satisfied before you can expect to transform any job into this ideal, it is surprising how many jobs can be rewarding when you approach them in the right way.

Job Satisfaction

In discussing job satisfaction, we must first answer the question: "Satisfaction for whom?" One would expect the characteristics of a satisfying job to be different for the person doing the job and for the person for whom it is done. However, there is one condition that is common: the job must have been a success. This, of course, leads to the next question: "How do managers and developers view project success?"

This question has been researched by Kurt Linberg [Linberg 99]. He studied the views of developers about project success for his PhD dissertation. In this study, he asked groups of developers to identify the most successful and least successful jobs on which they had worked. Then he examined the characteristics of the most successful and least successful projects.

Successful 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

From the developers' perspectives, the most successful projects had three common characteristics:

- The project was a technical challenge.
- The final product worked in the way that it was supposed to work.
- The team was small and performed well.

Some typical developer views about these successful jobs were that

- the product was well designed and implemented
- it was well tested
- the team members enjoyed working with each other
- they didn't feel pressured by management

In addition, as long as the project manager protected the team, conflicts with other groups or with senior management didn't seem to bother the team members. They also thought that the work was innovative, that there was a high level of trust among the team members and with the project manager, and that the team was highly motivated.

In general, the developers attributed their high levels of motivation to five things:

1. a sense of making a contribution
2. an orderly and well-managed working environment
3. frequent celebrations of even small successes
4. positive feedback from both management and marketing
5. the autonomy to do the job in the way that they thought best

It didn't seem to matter how important the project was to the organization or how well the team worked with other groups. In fact, it didn't even matter whether the job was completed on time and within its budget.

Unsuccessful Projects

Conversely, the team members viewed the least successful projects as unrewarding, as having poorly defined requirements, and as having inconsistent or even nonexistent marketing support. Typically, these worst performing projects were perceived as

- not achievable from the outset
- under excessive management pressure
- requiring unreasonable levels of overtime
- technically frustrating
- having frequent conflict among team members
- operating in a chaotic environment

In summary, the developers' views of project success were largely independent of cost or schedule performance. The team members viewed a project as successful if it was a rewarding and enjoyable experience, even if it was late and over budget. In other words, a successful project was a personally satisfying project.

Management's Views of Project Success

Linberg also found that management's views of project success or failure almost entirely concerned cost, schedule, and quality performance. Table 1 summarizes his findings on the definition of various levels of success or failure for both completed and cancelled projects [Linberg 99]. This study found that developers view some projects as successful even when management does not and vice versa. Having been a manager for many years and having seen how hard developers worked to meet their schedules, I thought this conclusion contradicted everything I have seen in more than 50 years of development experience. I have worked with hundreds of teams and supervised groups of thousands of developers who worked around the clock to meet their deadlines. Every team I have ever worked with has always made an extraordinary effort to meet

cost and schedule commitments, and many of them were consistently successful in doing so.

Project Outcome	**Completed Projects**	**Cancelled Projects**
Failure	A product that causes customer discontent: Not meeting user expectations.	Not learning anything that can be applied to the next project. Time and money wasted with no return.
Low Success	Below average cost, effort, and schedule performance. Barely meeting user expectations.	Learning can be minimally app-lied to future projects. Time and money wasted for limited return.
Success	Average cost, effort, and sche-dule performance. Meeting user expectations.	Learning can be applied to future projects. Some artifacts can be used.
High Success	Better than average cost, effort, and schedule performance. Fully meeting all user expectations.	Substantial learning can be app-lied to future projects. Significant number of artifacts can be used.
Exceptional Success	Meeting or exceeding all user quality, cost, effort, and schedule expectations.	A cancelled project cannot be called exceptionally successful.

A little reflection, however, shows that Linberg's conclusions do not conflict with my experience. I never asked the teams how they viewed their projects. They were paid to meet management's goals, and when they did, management was happy and everyone got paid. So, in summary, management views a project as highly successful if a team delivers a quality product on schedule and within its committed costs, regardless of how the team feels about the job. Conversely, development teams view projects as highly successful if they are professionally challenging and technically successful and if

they provide a rewarding and personally satisfying working environment.

Common Management and Team Goals

While no rational managers would object if developers enjoyed their work, that is not one of the managers' highest priorities. Similarly, developers universally would like to deliver quality products on schedule, and they even strive to do so, but they do it because they know it is their job. What they really want, however, is to have a rewarding experience.

While these views don't necessarily conflict, they are not mutually supportive. This situation is what is called *cognitive dissonance*, where our views of reality differ from what we experience. In the case of a project, the developers know what success feels like. Many of them have experienced it on sports teams where a winning performance is an exhilarating experience, the coach congratulates the players, and the fans cheer. Everybody then helps in the celebration.

But development is different from competitive sports. Developers know that the managers' kind of success is what the organization wants, and that what developers value most is not a management priority. Sometimes the team wins and nobody seems to care, and other times, it feels as if the team loses but management cheers. Cognitive dissonance is demotivating and debilitating; it is not the stuff that builds winning teams or ideal jobs.

Congruent Management and Team Goals

This raises the next question: "What would happen if the managers' and developers' views of project success reinforced and supported each other?" One team's experiences illustrate what could happen. On their last project, members of this team had put in more than 70-hour weeks, worked under enormous pressure, and managed to deliver their product to test on time. Testing, however, took longer than anyone had expected, and the product was delivered late. It was delivered, however, and at the time I met with the team, it was being installed by the users. While this job wasn't a complete failure in Linberg's terms, it certainly wasn't a big success from either management's or the team's perspective.

On the next project, however, the team members had a realistic plan; they did high quality work; they had capable leadership; they had clear, agreed-upon goals; and they believed that the team owned the project. This was their job and they were going to make it a success both for the team and for the business. They not only delivered their product to test on time, but it was of such high quality that testing was finished early. They only worked 45-hour weeks and were home for dinner with their families every night. They said it was the best project they had ever worked on, and management was full of praise.

How did they do it? The answer is that they and their management made a real effort to make this project a success for both the developers and the managers. More on how to do this in the June column.

Acknowledgments

First, I want to thank Bob Schaefer for his note. He asked about ways to handle conflicts between managers and developers and, in thinking about the answer, I got the idea for this series of columns. Also, 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, and Bill Peterson.

In Closing, an Invitation to Readers

In this particular series of columns, I have discussed some of the development issues related to large-scale projects. As I now consider the topics to address in subsequent columns, I would appreciate any of your comments and suggestions. If there are topics that you feel are particularly important and would like to see covered, please drop me a note with your comments, questions, or suggestions. Better yet, include a story or brief anecdote to illustrate 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

Reference

[Linberg 99]
Linberg, Kurt R. "Software Developer Perceptions about Software Project Failure: a Case Study," *The Journal of Systems and Software*, 49 (1999) 177–192.

About the Author

Watts S. Humphreyfounded 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 Architecture: The Next Generation

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, 2007

Robert L. Nord and Paul C. Clements, with help from members of the International Federation for Information Processing (IFIP) Working Group 2.10 on Software Architecture and the Working IEEE/IFIP Conference on Software Architecture (WICSA) 2007 working session on this topic.

The study of software architecture involves the understanding of the large-scale structures of software systems. As a field of study, software architecture is a specialization of software engineering and overlaps and interacts with the study of software families, domain-specific design, component-based reuse, software design, specific classes of components, and program analysis [Shaw 2006]. Beyond the related areas in software engineering, software architecture also interacts with activities in system architecture, information architecture, and enterprise architecture.

Since the mid 1980s, when the field began, (although its roots can be traced to earlier concepts), software architecture has largely been concerned with engineering and understanding the interaction of well-defined components over well-defined communication paths via well-defined interfaces. However, a much different world is emerging. Systems are built from components designed and produced by organizations independently of each other. The components may or may not have well-defined interfaces. The functionality and quality attributes of the total system may be derived from (rather than engineered into) the federation of components. The components and their interrelationships may change dynamically, arriving and departing either bidden or unbidden, and connecting to the part of the system where they will do the most

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 »

good. “Time to market” will be replaced by “time to useful functionality” and will be measured in seconds rather than months.

What will be the role of architecture in that new world? And what will the role of the architect be? How can the study of a system’s software structures have meaning if those structures are not known before execution and can change by the minute? It seems clear that the current architectural paradigm—focusing on the components and connections that make up a large-scale software system—will not suffice and will have to be replaced with something different. Nevertheless, it also seems clear that highly dynamic systems must be based on architectural principles that facilitate or guarantee system properties, thus preserving the need for software architectural thinking.

Members of the International Federation for Information Processing (IFIP) Working Group 2.10 on Software Architecture recently held a discussion on this topic and then continued the discussion during a working session at the Sixth Working IEEE/IFIP Conference on Software Architecture (WICSA 2007).

IFIP Working Group 2.10 on Software Architecture

IFIP is a non-governmental, non-profit umbrella organization for national societies working in the field of information processing. It was established in 1960 under the auspices of UNESCO as a result of the first World Computer Congress held in Paris in 1959. IFIP maintains several dozen working groups that strive to advance the state of and foster international cooperation in their respective areas.

The purpose of IFIP Working Group 2.10 is to further the practice of software architecture by integrating software architecture research and practice. The group sponsors the Software Architecture Portal, which provides information about IFIP WG 2.10, the WICSA conference series, the Software Architecture Village Wiki, and other resources on software architecture.

Members of the group are Maarten Boasson, Jan Bosch, Paul C. Clements, Morven W. Gentleman, Rich Hilliard, Christine Hofmeister, Philippe Kruchten, Tomi Männistö, Robert L. Nord, Henk Obbink, Alexander Ran, Mary M. Shaw, Judith A. Stafford, Hans van Vliet, and Eoin Woods.The working session had two goals:

1. to lay out certain dimensions of the problem space that we assume now but that may vanish in the next 10 years
2. to define a broad description of the practice of software architecture and the role of the architect in the most demanding part of that new problem space

The working session was devoted to trying to lay out the space of the practice of software architecture. The purpose was to define two points: a point where the field is now in that space and a point in the same space where the field might be in x years. The group held a brainstorming session in which participants suggested a dimension of the space and what it means to be at the most primitive and most advanced points along that dimension (which we arbitrarily labeled 0 and 10, respectively).

The working session’s 22 participants created these 12 dimensions of the space of architecture:

1. *codification and socialization*—processes by which an architect communicates architectural ideas to stakeholders. *Codification* refers to specification of the architecture, while *socialization* refers to the less formal processes by which the architecture is internalized. Socialization can happen through conversation, training, and so forth. Codification and socialization are complementary processes; they should enforce each other.
   - 0—Codification and socialization do not occur.
   - 10— Codification and socialization are in balance with each other; socialization becomes a process for codification; socialization and codification enforce each other and work in a global environment.
2. *handling quality attributes*
   - 0—There is no consensus on quality attributes.

- 10—Quality attributes are linked to business and engineering needs and are quantitatively specified.

3. *architectural automation*
   - 0—There is no support for architectural automation; description consists of human language on paper.
   - 10—Language for architecture is the base language for automation.
4. *architectural specificity*
   - 0—Reasoning is limited to specific elements (hardware, network, software, etc.) and the relationships known by the architect.
   - 10—Architects employ reasoning about guidelines, constraints, quality attributes—self-adaptive systems.
5. *architectural responsibility*—degree to which an architect is responsible
   - 0—There is no explicit recognition of responsibility. The worst case is having responsibility and no authority.
   - 10—A clear definition of responsibility and authority exists for the architect's role. Responsibility is sufficient to deliver the function and quality attributes of the system to its stakeholders over time.
6. *accidental versus intentional architects*
   - 0—Architects have no explicit training, no career path, no formal explicit recognition, no experience threshold (relating problems to domain, development, implementation, failed, organization, etc.).
   - 10—Architects are chosen intentionally for their ability.
7. *domain versatility and adaptability*
   - 0—The architect has a one-track mind.
   - 10—The architect is pragmatic and inquiring—able to organize information.
8. *architecting process*—maturity of architectural choices
   - 0—Diverse solutions and techniques exist, but the choice is arbitrary.
   - 10—There is a clear linkage between architecture goals and the choice of process and techniques.
9. *technology dependency*—Technology is a tool.
   - 0—Architecture is constrained by the existing technology.
   - 10—Architecture is fearless, specifying the abstract solution without necessarily any bindings to existing technology. Architecture is not constrained by existing technology. A fearless architecture is one that might influence future technology.
10. *creating versus choosing an architecture*
    - 0—No previous solutions are reused.
    - 10—The solution already exists (not accounting for the details) —looking at previous solutions and reusing them.
11. *complexity*
    - 0—Complexity is low.
    - 10—Architecture is produced for complicated and complex systems with emergent behavior.
12. *interdisciplinary architecture*

0—A single discipline is fully understood (e.g., information architecture, enterprise architecture).

- 10—Multiple disciplines are fully integrated. Stakeholders' perspectives are met.

Later, the group postulated the state of the practice in each of the dimensions and what it would take to move it to level 10. Time limits allowed the completion of only the first three areas. These details are available at the conference Wiki. Readers of this column are encouraged to contact the authors or to log on to the conference Wiki to either provide feedback on the proposed dimensions or propose additional ones.

Working IEEE/IFIP Conference on Software Architecture (WICSA)

The mission of the WICSA conference series is to be the premier means of communication and advancement of research and practice in software architecture, from both academia and industry, worldwide. Working sessions have an important role at WICSA in fostering this dialog. An online record of the discussion from the previous three conferences is captured in the conference Wiki.

Plans are underway for WICSA 2008, to be held in Vancouver, Canada, February 18-21, 2008.

Reference

[Shaw 2006]
Shaw, M. & Clements, P. "The Golden Age of Software Architecture." *IEEE Software 23*, 2 (March/April 2006): 31-39.

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).

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.

Share This Page

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

Workshop to Present Best Practices in Software Architecture

NEWS AT SEI

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, 2007

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 »

"For the highly complex, massively large-scale, and highly interoperable systems that we need now and in the future, architecture is the linchpin," Rolf Siegers of Raytheon said at last year's Software Architecture Technology User's Network (SATURN) workshop held by the Software Engineering Institute (SEI). To support that statement, he cited research by Barry Boehm and Richard Turner that examined over 161 software-intensive system projects ranging from 2,600 to 1.3 million lines of code [Boehm 2004]. The results showed that increased investment on time spent architecting resulted in significantly less system rework and shorter time to market. "We all agree that this discipline [developing architectures] is a better way to do things, but having quantified results makes it far easier to establish it as part of your business case."

Over the past year, what Siegers said hasn't changed. So the SEI is continuing to help organizations build a business case for architecture and fine-tune how they can produce high-quality systems through its SATURN workshop—a yearly gathering of users of SEI architectural methods and techniques. At SATURN, engineers, architects, and technical and product managers exchange their best practices for and personal experiences in developing or acquiring software architectures and using them to build predictable, successful systems. SATURN 2007 is scheduled for May 14 to 16 in Pittsburgh.

During his keynote at SATURN 2006, Siegers shared Raytheon's experience using SEI methods such as the Architecture Tradeoff Analysis Method (ATAM) evaluation and Quality Attribute Workshop (QAW). He explained why his company has incorporated ATAM evaluator training by the SEI as an explicit requirement in its certified architect program: "We adapted the ATAM to systems architecture to address a need that couldn't be met by anything else out there—ATAM filled a void that no one else touched." The Raytheon Certified Architect Program (RCAP) also uses the QAW to

help clarify nonfunctional requirements when necessary.

The ATAM and QAW are also standard methods in the architecture toolkit that Don O'Connell draws from in his work at Boeing Phantom Works. During his SATURN 2006 keynote, O'Connell described his experience using ATAM this way: "ATAM evaluations that focus on cost reduction enable us to identify and mitigate risks—a process that reduces rework and therefore cost."

Such real-life ways of using SEI technologies are exactly what draw people to SATURN workshops. O'Connell finds SATURN unique because of its interactivity. "Instead of just listening to presentations, SATURN provides opportunities to learn where people are actually struggling to find solutions to problems," he said.

The numbers of organizations participating in architecture-centric conferences and workshops such as the Working IEEE/IFIP Conference on Software Architecture (WICSA) [WICSA 2007] and SATURN [SEI 2007] are growing each year, showing increased interest in the value of software architecture. A total of 38 organizations took part in the first two SATURN workshops, and at least 25 more will be joining those ranks this year. SATURN 2007 has the most international involvement so far, with presenters from Korea, India, Australia, and Ireland.

Proof that architecting is becoming more valued is also evidenced by the recent efforts of large organizations such as Raytheon to establish a variety of architecture initiatives such as certification programs to improve and support the company's architecture practices. Other organizations such as Siemens and Bosch show a similar dedication to creating high-quality software with their well-established research centers that focus on architecture practice. Additional organizations such as Unisys have joined the ranks of those committed to using the ATAM as standard practice. All of that evidence led workshop co-chairs Ipek Ozkaya and Rob Wojcik (both of the SEI) to oversee the creation of a diverse program for SATURN 2007. In addition to focusing on best practice and experience sharing, this year's workshop will highlight these and other topics:

- how to clarify and codify the role of the software architect
- the economics-driven aspects of software architecture
- the relationship between software and system architecture
- how to improve the state of architectural practices across an organization
- architecturally significant requirements
- how to apply SEI software architecture methods within organizations

The SATURN 2007 program also includes

- a panel session on experiences using the ATAM
- tutorials that cover topics such as evaluating service-oriented architecture and integrating architecture-centric methods into object-oriented analysis and design
- working sessions on economics-driven architecting and architecture-based system evolution

Two keynotes are planned by these software architecture experts:

- Ian Gorton of Pacific Northwest National Laboratory (PNNL) will describe some novel, prototype tools for architecture knowledge management, collaborative architecture design and decision making, and performance analysis of COTS-based architectures.
- Jeromy Carrière of Fidelity Investments will present a retrospective that journals his experiences practicing architecture in environments as varied as telecommunications, academia, dot com and post-dot com era startups, a large independent software vendor, and enterprises in media and financial services.

SATURN pre-registration closes at 5 p.m. Eastern time on May 9. After that, all SATURN attendees may register on-site. For more information about the SATURN 2007, go to http://www.sei.cmu.edu/saturn/.

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

[SEI 2007]
Software Engineering Institute. *Third SEI Software Architecture Technology User Network Workshop*. http://www.sei.cmu.edu/saturn/ (2007).

[WICSA 2007]
IFIP Working Group. *WICSA: The Working IEEE/IFIP Conference on Software Architecture*. (2007).

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

CMMI: Beyond V1.2 Workshops

NEWS AT SEI

Author

Mike Phillips

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

Process Improvement

CMMI

This article was originally published in News at SEI on: April 1, 2007

My previous column introduced a new component of the CMMI Product Suite: *Understanding and Leveraging a Supplier's CMMI Efforts: A Guidebook for Acquirers*, otherwise known as the Acquisition Guidebook. This new guidebook is designed to help government acquisition organizations use the existing CMMI Product Suite to select qualified suppliers. In this month's column, I'll describe the work currently under way by the CMMI Team—government, industry, and SEI—to seek ideas for further improvements in the CMMI Product Suite over the next several years.

Background

The initial launch of the CMMI Product Suite (V1.0) was based on the need to integrate three process-improvement models and their associated appraisal methods and training into one integrated package. The second release of CMMI (V1.1) was a planned but moderate update of the product suite in response to the feedback from initial users. This release consisted primarily of refinements to the Standard CMMI Appraisal Method for Process Improvement (SCAMPI^{SM}) appraisal method. SCAMPI was moved from a discovery-based method to a verification-based method to reduce the amount of time spent on site and to enable easier evaluation by external (government) sponsors either seeking or executing a contract. The recent V1.2 release was based on four years of feedback collected from CMMI users, which included feedback mechanisms such as direct change requests from individuals and groups and an SEI project called Interpretive Guidance that helped guide the kinds of change made in V1.2 to better meet the needs of the broad set of users of the CMMI Product Suite.

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

The co-sponsors of CMMI are the Office of the Secretary of Defense for government and the National Defense Industrial Association (NDIA) for industry. These sponsoring organizations determined that the CMMI project team should investigate the next steps for the next release of CMMI earlier than in the past. This early investigation recognizes the need for research and experimentation on alternatives for growth as the product suite becomes more and more widely used as a process-improvement tool. A series of workshops is now under way to begin the search for both incremental improvement and major innovation. The NDIA is leading these workshops, and the SEI is providing technical support (a similar sponsorship arrangement to the one used for the annual CMMI Workshop held every November in Denver, Colo.).

As of this writing, the first two workshops have been held—one in San Francisco and one in Washington, D.C. Three international workshops are planned in the summer and fall. The next will be held in Montreal, Canada, July 10 to 13. Two others are planned: one to be held in London, U.K., October 9 to 12, and the other to be held in Sydney, Australia, November 27 to 30. Additional U.S. and international workshops may be considered in 2008.

Workshop Approach

The approach of these workshops has been to encourage innovative thinking about how the CMMI Product Suite should evolve in its second decade of use. The workshop agenda is divided into two segments: two days on the model aspects of CMMI and two days on the CMMI appraisal methods. While all workshop participants are invited to attend the first segment, some experience with SCAMPI or related process-improvement methodologies is requested for attendance at the appraisal-related segment. A set of starter questions is provided to stimulate discussion at each workshop, and new questions are added that are pertinent to the specific interests of each workshop.

The set of starter questions is:

- Do we need something different or additional to define CMMI high maturity (i.e., CMMI levels 4 and 5)?
- How can we reduce the size of CMMI models while preserving integrity? Can we likewise reduce the size of the appraisal method?
- Can we eliminate the staged representation?
- Is the CMMI V1.2 constellation strategy the right approach?
- Can we identify the next-generation process-improvement methodology?
- Can CMMI be harmonized with other continuous-process-improvement approaches?
- Can repeatability, consistency, and the overall model and appraisal methodology be improved?
- Are there breakthrough concepts that we can apply to overallprocess improvement?

Initial Thoughts After Two Events

The discussion in each of the first two workshops brought together a mixture of government, industry, and academic perspectives. A summary of the diversity of opinion would not properly recognize the many innovative ideas that were recorded in these workshops to assist CMMI efforts over the next few years, but the following glimpse of the ideas, issues, challenges, and opportunities can give you an idea of the input we've received so far.

From a model perspective, the efficacy of the constellation approach remains uncertain in the minds of workshop attendees. Since none of the new constellations is fully matured, the discussions were often about how the various constellations might be used together. Many attendees were from organizations responsible for development and at least one of the other areas covered by the new constellations in our plans for V1.2 release. Since the intent of CMMI is to avoid a proliferation of improvement models, do the constellations help or hinder that strategy? How can organizations avoid duplicating appraisal effort when seeking to address a mixture of projects when

(< 5 minute) survey.

some are strongly flavored with one emphasis (e.g., development) while others are equally strong in their focus on another (e.g., acquisition)?

Another area that prompted extensive discussion was process performance. In a 2004 *Crosstalk* article, Dr. Robert Charette, Laura Dwinnell, and John McGarry wrote that a wide-ranging set of appraisals, conducted as part of the Tri-Service Assessment Initiative, found many deficiencies in the process performance of more than 90% of the DoD programs examined [Charette 04]. While about half of these programs had fundamental problems of process adherence that CMMI readily addresses, an additional dimension of deficiencies involved program teams—often across multiple companies—not having adequate tailoring of various organizational standard processes to meet the challenges of complex software-intensive development by the full team. While CMMI already encourages senior management to review the results of process-improvement activities, senior management may need to increase its focus on process effectiveness and efficiency and on the appropriate tailoring taking place in program teams.

Another theme worthy of note is the need to relate CMMI to the many other standards and methodologies that organizations often must consider for a variety of reasons. In previous columns I have written about the compatibility of CMMI with improvement approaches such as Six Sigma and development approaches such as the SEI Team Software Process (TSP) methodology. Many workshop attendees suggested ways to relate the various approaches together to capture their synergy without demanding that any particular coupling be accomplished. A simple example was that terminology from related standards (e.g., IEEE 12207 or 15288) might be chosen for future updates to minimize unintended differences in interpretation for similar practices.

On the appraisal side, workshop attendees were interested in reducing the impact of appraisal preparation on the organization undergoing the appraisal. While the move from a discovery-based to a verification-based methodology in the SCAMPI method does reduce on-site time, it places a higher burden on the organization gathering the practice implementation indicators (PIIs). Some participants asked if there could be ways to reduce this burden without sacrificing appraisal integrity, repeatability, or consistency.

A longstanding discussion continues in the workshops about the advantages and disadvantages of having the choice of two representations for an appraisal. Attendees believed that the distinctions between the representations have been reduced over time, most notably by the single-book approach and the elimination of advanced practices and common features.

A final theme that couples both model and appraisal issues is to view the model in a new way. This view divides model practices into two fairly distinct groups. One group emphasizes doing the work. These practices are well documented in the Project, Engineering, and Support process area categories at maturity levels 2 and 3. The other group emphasizes the organizational elements and the higher-maturity elements in which the practices focus on process improvement. These two parts mix nicely in use and appraisals, but it might be possible to focus on best practices for doing the work in one appraisal and focus on process improvement in another type of appraisal.

Summary

All of the ideas gathered in these workshops will take time to investigate and pilot—so our aspiration to release a new version of the product suite every 3-5 years does not appear to need adjustment at this point. I encourage all of you to consider attending one of the upcoming workshops to contribute your ideas. If you cannot attend, you can still submit your input and ideas. To ensure that the right groups receive your comments, please complete and submit a change request form.

I will be providing updates of the progress of these workshops both in future columns and at a variety of conferences over the next year. The CMMI Product Team intends to continuously improve the CMMI Product Suite to provide greater value for a growing population of CMMI users.

In my next column, I will discuss the work we have been doing to continuously

improve V1.2 appraisals. While much has been done in the current product-suite release, our new approach to the certification of SCAMPI Lead Appraisers will leverage the current capabilities of Lead Appraisers while helping them to build their professionalism.

References

[Charette 04]
Charette, Robert; Dwinnell, Laura; & McGarry, John. "Understanding the Roots of Process Performance Failure." *Crosstalk 17*, 8 (August 2004): 18-22.

About the Author

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.

Challenges of Establishing Network-Centric Operations I: Technical Research Challenges

NEWS AT SEI

Authors

Grace Lewis

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, 2007

Government and commercial organizations have initiated large-scale efforts to migrate rapidly toward network-centric operations. This involves moving and sharing information in an agile manner among personnel and systems with the network as a central enabling mechanism. This series of columns identifies challenges of developing network-centric operations related to three specific aspects of the migration: technology challenges, organizational challenges, and programmatic challenges within government organizations. This column focuses on technology challenges—specifically on those related to service-oriented architecture (SOA), a key enabling technology for migration to network-centric operations.

We focus on the SOA paradigm because it is having a substantial impact on the way software systems are developed. Organizations as diverse as the U.S. Department of Defense, federal agencies, banks, and health care providers are focusing on SOAs as a way to achieve interoperability among systems and agility within business practices. Although significant progress has been made, current efforts have been evolving in many directions without a central focus. As a result, there is a danger that important research needs will be overlooked while efforts focus on issues of peripheral long-term interest.

We have been addressing the need for a long-term SOA research agenda through an SEI independent research and development project that leverages the thinking of an international team of academic and corporate researchers. The team has developed a

Related Links

Training

Migrating Legacy Systems to SOA Environments - eLearning

Service-Oriented Architecture: Best Practices for Successful Adoption

See more related courses »

draft research agenda with a set of research challenges related to business, operations, engineering, and cross-cutting concerns. This research agenda helps to sensitize practitioners to critical unsolved problems and to focus researchers on the highest-payoff issues. We hope that the research agenda will lead to the establishment of a longer-term community of interest.

Team members have presented this draft research agenda at workshops at three international conferences—the Conference on Software Maintenance and Reengineering (CSMR 2007), Interoperability for Enterprise Software and Applications (I-ESA 2007), and the International Conference on Software Engineering (ICSE 2007), as well as at Canada's Consortium for Software Engineering Research (CSER). At each of these conferences, participants discussed the papers presented in terms of how they contribute to the research agenda. The research agenda is currently being refined based on input from these forums.

The rest of this article highlights at a high level a selection of the research challenges of the solution space. A more detailed version of the draft research agenda can be found in the conference proceedings [Kontogiannis 07]. The revised research agenda will be published as an SEI technical note in Fall 2007.

Context for SOA Research Agenda

Service orientation cuts across the enterprise and traditional domains. Because different groups focus on different aspects of SOA, a comprehensive research agenda requires factoring in the perspectives and needs of groups across the enterprise. Software engineers focus on functional requirements, components, integration techniques, messaging, tools, development environments, and middleware. Business people focus on implementing business strategies, enabling leaner IT departments, facilitating agile process models, and driving new products. Operational users are concerned with transparency, flexibility, ubiquitous access to services, and applications that ease their lives (e-government, e-health, entertainment).

Figure 1 illustrates the context for understanding SOA research issues. The top part of the figure illustrates an idealized SOA-adoption setting in which an organization's business drivers, context, and application domain frame the problem space. The middle part of the figure illustrates the development of a service strategy. A service strategy defines the goals and objectives for adoption of SOA as well as the development of an SOA strategy that takes into account business and engineering drivers. The bottom part of the figure illustrates the solution space that requires decisions in the domains of engineering, business, and operations, as well as cross-cutting issues. The research agenda focuses on challenges in the solution space.

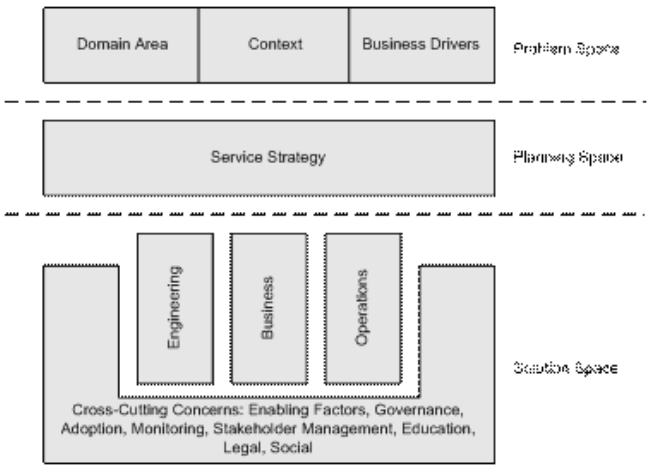

Figure 1: Context of SOA Research Agenda

Business Challenges

The business domain deals with the form and the impact that service orientation may take in a given organization, application domain, or context. Drivers for service-oriented systems in this domain include the need to be on demand, customizable,

trusted, compliant, agile, and measurable. Figure 2 illustrates the major categories of business issues.

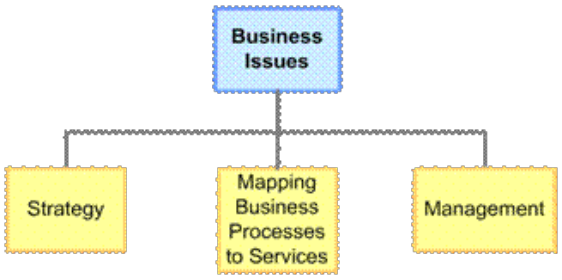

Figure 2: SOA Business Issues

Business-related research issues include

- strategy
- techniques to establish and document the business case for service orientation
- techniques for strategy selection—evaluation of fit of strategy with current and future enterprise planning; strategic business process reuse, operations support; best match for business drivers; domain and context
- mapping business processes to services
- techniques for service identification—methods and models for the selection of appropriate services for a given domain
- techniques and processes to support the strategic reuse of legacy components and services
- analytic methods for service evaluation—how to evaluate if a service fits certain business needs
- techniques and processes for establishing relations between business models and service models
- investigation of industry and domain-specific standards for business and service modeling
- processes to support the adaptation of services to meet changes in business processes
- management
- models for workforce allocation—alignment of people across organizations to implement agile business processes, collection and analysis of data for workforce management and productivity optimization
- investigation of models for contract pricing and negotiation in an on-demand service setting

models for organizational structures in SOA environments

- patterns for roles and responsibilities
- methods for assessing the effectiveness of services

Operations Domain

The operations domain deals with the operation, evaluation, and optimization of service-oriented systems. Operational drivers include the need for service-oriented systems to be ambient, user friendly, high impact, pervasive, and adoptable. This domain includes issues of monitoring, adoption, and support of deployed service-oriented systems. Figure 3 illustrates operations issues at a high level.

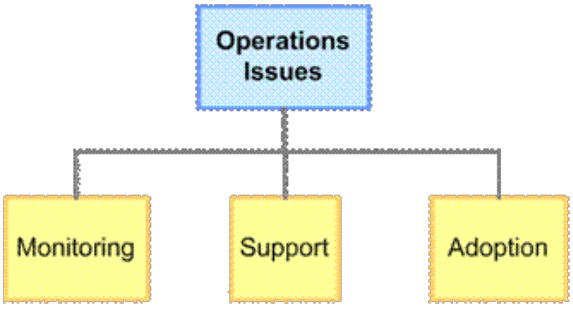

Figure 3: SOA Operations Issues

Research challenges of SOA operations include

- monitoring
- techniques for monitoring of business processes in an SOA environment
- monitoring of SOA operations—self-healing systems, instrumentation, diagnostics, logging, dynamic configurations of large-scale systems
- techniques for resource allocation and configuration management in an SOA environment
- support
- techniques and methods for problem management in an SOA-enabled organization
- techniques and models for the specification and dissemination of service-level agreements
- adoption
- investigation of requirements for portals and collaboration environments
- analysis and models of customer-adoption processes
- investigation of methods for evaluating and assessing service usability

issues and models related to the adoption and advertisement of services

Engineering Domain

The engineering domain deals with the engineering aspects of the service-oriented system life cycle. Some of the driving characteristics for service-oriented systems in this domain are reliability, security, openness, robustness, efficiency, and testability. Issues include process models that can be used to build service-oriented systems, requirement models for denoting functional and non-functional aspects, platform- and computational-independent architectural abstractions, design patterns, logging, model-driven code generation, verification, testing, and maintenance. Figure 4 illustrates the high-level engineering issues.

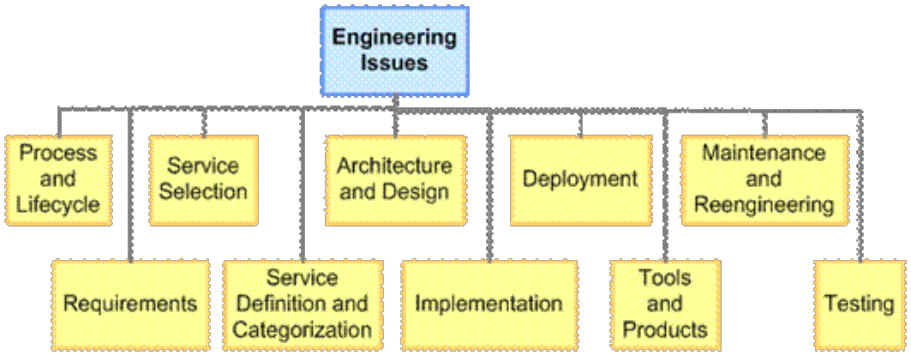

Figure 4: SOA Engineering Issues

Selected engineering research issues include

- process and life cycle
- development processes and methodologies for service-oriented systems
- service-oriented system lifecycle issues—managing a continuous running system in a distributed setting with distributed and diverse stakeholders
- requirements
- modeling and management of non-functional, soft, evolvable requirements—conflicting requirements, usability, adoption, adaptability, continuous evolution
- requirements specification, modeling, and management from the unique perspectives of service provider, service consumer, and infrastructure developers
- service selection
- service-selection criteria—reliability, performance, risk
- service-evaluation techniques
- models to support strategic reuse
- service definition and categorization
- taxonomy and repositories of best practices for the definition and classification of services

- guidelines for design decisions—e.g., appropriate granularity of services to be used
- techniques and models for the syntactic and semantic description of services
- architecture and design
- modeling of the dynamic runtime architecture of an SOA system
- features and properties for next-generation frameworks for development of service-oriented systems
- architectural styles for service-oriented systems
- architectures for data integration in service-oriented environments—mediation, consolidation, ownership, semantics, metadata
- communication/connectors—synchronous, asynchronous
- styles and design decisions that favor certain non-functional requirements—design for security, performance
- architectures for service types
- data services (information as a service)
- business services
- infrastructure services
- design patterns for service-oriented systems
- design for personalization, context awareness, and adaptation—time zones, language
- design for runtime semantic-based discovery and composition
- relationship between product lines and service-oriented systems—services as core assets, variability points for service providers
- protocol mediation and wrapping in multi-protocol environments
- implementation
- model-driven approaches—platform-independent models, platform-specific models, template-based code generation
- language extensions to support service-oriented development

- exception handling
- tools and products
- integrated development environments to support service-oriented development
- tools for logging, monitoring, and diagnostics
- evaluation guidelines for tool and product support
- support for service choreography and orchestration
- collaboration tools in distributed-development environments
- testing
- infrastructure testing—messages, specifications, model properties, deployment descriptors
- functional service testing—white-box and black-box testing of services that consider all potential compositions
- integration testing—transaction management, quality of service, load/stress testing, composition, workflow simulators, orchestration, versioning, monitoring, and regression testing
- system testing in dynamic environments
- simulation and what-if analysis testing
- acceptance testing—possibly more diverse that typical acceptance testing
- practices for service providers to support system testing of their consumers—parallel services for testing, test cases, test data
- establishing testbeds and benchmarks
- deployment
- pre-start checks to ensure that all required components and configurations are properly set
- techniques for multi-platform support configurations
- techniques for context and location-awareness support
- maintenance and reengineering
- evolution patterns

- dependency and impact analysis
- infrastructures for change control and management
- tools, techniques, and environments to support maintenance activities
- multi-language system analysis and maintenance
- reengineering processes
- tools for the verification and validation of compliance with constraints
- round-trip engineering in service-oriented systems
- resource allocation

Cross-Cutting Concerns

There are a number of issues that have an effect on all domains. Examples of such issues are governance, training and education, social and legal issues, and people skills. Figure 5 illustrates the high-level cross-cutting issues.

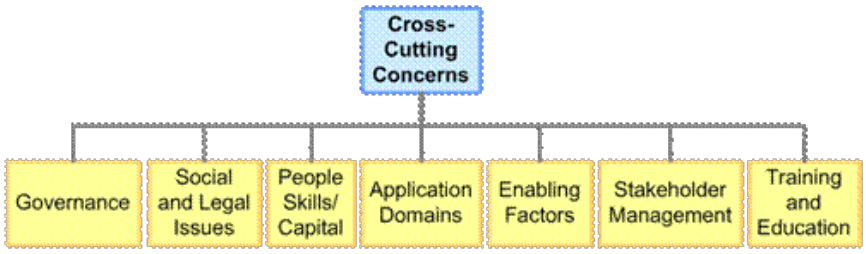

Figure 5: SOA Cross Cutting Concerns

Selected cross-cutting issues include

- governance
- techniques and processes to model policy, risk, and trust and to ensure that a service acts on requests that comply with claims required by policies
- investigation and compilation of repositories and guidelines of best practices for compliance monitoring in given domains
- social and legal issues
- social and legal issues related to the deployment and use of services
- legal compliance
- roles of service orientation in social computing
- people skills/capital

- skills required to develop, use, and maintain a service-oriented system
- matching needed services with user skills and abilities
- application domains
- investigation of application domains such as health care, government, finance, banking, electronics, telecommunications, and automotive, and potential impact
- establishing industry- and domain-specific standards for data, services, business processes, best practices, and performance indicators
- enabling factors
- analysis of technology issues as an enabler for service orientation
- stakeholder management
- techniques to mediate different and diverse stakeholders' needs, requirements, and views
- training and education
- establishing the area of services science
- investigating and developing appropriate university curricula

Conclusions

In this overview, we provide an initial classification of research issues in three domains —business, engineering, and operations, plus a set of cross-cutting concerns. We believe that the SOA approach will have a significant impact on software engineering in the future. However, there must be a clear understanding of what it can and cannot do as well as a roadmap for its future evolution.

The next steps for this work are to obtain additional feedback from research and practitioners on the classification of research issues and their validity. After we process this feedback, we will develop a more detailed research agenda to help guide both practitioners and researchers and to help to lead to the establishment of a longer-term community of interest.

References

[Kontogiannis 07]
Kontogiannis, K; Lewis, G.A.; Smith, D.B.; Litoiu, M.; Müller, H.; Schuster, S.; & Stroulia, E. "The Landscape of Service-Oriented Systems: A Research Perspective. SDSOA 2007: International Workshop on Systems Development in SOA Environments." In Proceedings of International Conference on Software Engineering (ICSE 2007), Minneapolis, MN: May 2007.

About the Authors

Dennis Smith is the lead for the SEI Integration of Software Intensive Systems (ISIS) 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.

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 more than 15 years of experience in software engineering. She is also a member of the technical faculty for the Master's in Software Engineering program at CMU.

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

Army Engineering Center Chooses SMART Approach to SOA Planning

NEWS AT SEI

Author

John Morley

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, 2007

One reason organizations are keenly interested in service-oriented architecture (SOA) is that it offers the promise of reusing legacy-system code as services that represent business tasks. Some common services are customer lookup, account lookup, credit card validation, and hotel reservation.

Recently, the Software Engineering Institute (SEI) introduced a new method for planning the migration of legacy components to services called the SEI Service Migration and Reuse Technique* (SMART) method. One of the first organizations to adopt SMART is the U.S. Army Communications-Electronic Research Development and Engineering Center (CERDEC).

news@sei spoke with SEI staff members Grace Lewis and Dennis Smith, who are leading the project to increase adoption of the SMART method.

What is SMART and what problem is it designed to solve?

Lewis: SMART is a technique for helping organizations understand in detail the effort it takes to migrate specific legacy components to a service-oriented architecture. When we do a SMART engagement, we look at code and say "You would have to do this to the code; you would have to do that to the code." Then, we get people to understand that what is required is more than adding an interface to the code.

Smith: SMART helps an organization get a very realistic sense—is it viable to migrate these components as services, which services make sense, and what kind and level of

Related Links

Training

Migrating Legacy Systems to SOA Environments - eLearning

Service-Oriented Architecture: Best Practices for Successful Adoption

See more related courses »

resources are needed?

Do you find that organizations think migrating legacy components to services is easier to do than it is?

Lewis: There is a lot of hype about service-oriented architecture, in part caused by vendors who tell organizations that it is something extremely easy.

Smith: A lot of the literature about SOA has been produced by vendors who sell products. So is it the accurate picture? We take a vendor-neutral approach: What about SOA has value and what does not have value. If you are a top-level manager, you hear a lot about leveraging legacy systems as services, but you probably don't know where to go next.

You are now transitioning SMART to the U.S. Army CERDEC.

Lewis: Yes, CERDEC is an Army transition program. The CERDEC job is to build systems and look at technology (which is what they are doing with SMART). When they understand the technology well, they transition it to other people in the Army.

Why is CERDEC interested in SMART?

Lewis: Our first engagement with CERDEC was about two years ago. Back then, CERDEC had been developing a system and wanted to know what it would take to move that system to a service-oriented architecture. CERDEC had two targets for service-oriented architectures, SOSCOE [System of Systems Common Operating Environment] and NCES [Net-Centric Enterprise Services]. We worked with the people at CERDEC—looking at their code, analyzing their architecture—and at the end of that we recommended that they wait. SOSCOE and especially NCES then were still evolving so that any effort at that point could have been wasted. Also, our customers were initially thinking that they would migrate the whole legacy system, not components of it, to a service-oriented architecture. But you actually need to find the specific set of components that it makes sense to migrate. We advised them to wait 18 months at least, to a time when SOSCOE planned a build. Eighteen months after our recommendation, we re-engaged with CERDEC at their request.

What has been done to transition SMART to CERDEC?

Lewis: First, we gave people at CERDEC a tutorial to help them understand what SOA is all about and gain an overview of SMART. The tutorial was well attended, by about 25 people. We have given this tutorial on the migration of legacy components to service-oriented architectures in several other settings. Then, we scheduled pilots. The first pilot was for them to watch us perform SMART. We repeated that pilot to make sure all of the key people saw us performing the SMART process. The second pilot will be for us to watch them conduct the SMART process. We will coach them on their use of the technique.

Smith: And at some point, we will check to be sure that we are comfortable with their understanding and that they are comfortable doing it. They want to be sure that they can facilitate the use of the technique. We may repeat this process if needed.

Lewis: After that, CERDEC will understand the method and be ready to apply it.

How does the SMART approach start?

Lewis: The first step in SMART is to establish the migration context. We have a set of questions about why organizations want to migrate, such as do they understand what a migration will entail and what are the business and technical drivers. We also want to know what the legacy system is about, at a very high level, what kinds of services they think they want to produce, and what are the types of service consumers.

Smith: Both the kind of services and the service consumers are important. As Grace pointed out earlier, many organizations initially want to turn a legacy *system* into services. In reality, they need to target more finely. They need to determine what specific services they want from a legacy system. It is also important to understand who is going to use those services. If they don't do this, they could run into a situation

thinking "if we build it they will come." It might be that nobody will come.

Lewis: At the end of this step, you reach a decision point: Do you have enough information to continue? If not, you need to get the answers to those questions before continuing with the process.

After an organization knows the services and service consumers, what then?

Lewis: Then you describe the system capability and the target SOA environment. In parallel to determining that information, you can start to map components to services.

Smith: The focus at this stage is on the specific components identified earlier. One goal of SMART is to find a relatively small number of components that you can target for turning into services. You home in on them, rather than plowing through the entire legacy system.

Lewis: The most common target is Web services with some enterprise bus-service product; this kind of environment we call “plain vanilla.” For CERDEC, SOSCOE is not a common target; it is built from scratch almost, so that will allow real-time systems to be created.

Smith: Most people equate SOA environments with Web services. And it is common, but it is not the only environment. And for SOSCOE, it is built from the ground up because it has hard, real-time constraints.

What if a customer doesn't have a target in mind?

Smith: We can do an SOA strategy workshop to help a customer find ways to make some of those decisions.

Lewis: It is important that organizations decide the target or targets, based on IT infrastructure. If they haven't identified the target, then they should not go ahead. This is something that SMART enforces.

After organizations identify the target, where does SMART take them?

Lewis: Our next step is to analyze the gap between where they are and where they want to go. To this point, SMART helps them understand their key legacy components, service consumers, and target environment. Now, SMART helps them look inside each component. What would a developer have to do to the code for that component to turn it into a service that meets the needs of the consumers? The involvement of the developer is critical here because the developer can say, “We must do this, and this, and this.” In this stage, the organization gets into a great deal of detail. The organization gains a truer understanding of how long it will take and how much it will cost.

It could also happen that people do not understand what they have—a legacy system could have been created long ago, and documentation is missing, if it existed at all. We can do some architecture reconstruction, some code analysis to try to understand basics about the legacy system. We can offer some services in doing this, if the code is written in a language where there are tools we can work with. Ideally, the developer is there; otherwise, we have to do some work to arrive at the information the developer can provide.

Smith: With SMART we try to take as little time as possible to get the most significant information to make decisions. We think that this approach saves customers money and makes the best use of our resources. If there are real gaps, we have to delve deeper, which costs more. At CERDEC, the people who developed the legacy components are still around. They really understand their code, so we have great confidence in their estimates of how long the changes will take.

What does SMART help an organization produce from all of the information?

Lewis: The final step in SMART is producing a migration strategy, which is a combination of a lot of things. This is important: The outcome of SMART is not the migrated code; the outcome is a strategy for people to follow to do the migration.

The strategy could be straightforward—as in "take the code and do exactly what you said you needed to do." Or it could be "define an architecture for your services and conduct a strategy workshop because you really don't understand what is needed."

Part of the migration strategy is to address what is known about the technologies that the organization wants to use. The SEI can offer the SEI T-Check[SM] method to investigate technologies. This method is a low-cost way to separate hype from reality about a technology as its use pertains to the target environment the organization has chosen. For example, if a part of the system will involve the transferring of images, the organization needs to be sure that its chosen technologies are capable of doing that.

Smith: There's also the SOA strategy workshop to help organizations understand the value of putting together a realistic approach for moving into the SOA world. This involves looking at technology in general and what they want to do with their legacy systems and infrastructure options, among other things. And we can also offer an SOA governance workshop.

How long does it take for you to teach SMART to an organization?

Lewis: It does depend on the situation. At CERDEC, our first pilot took one visit, and we spent some time to do a report—about six weeks in elapsed time.

Smith: Typically we expect about three visits of two to three days each with time for report preparation—six weeks to two months in elapsed time. We are developing an automated tool to streamline the interview process. The work in establishing the migration strategy, describing existing capability, and describing the target SOA environment is done through a service migration interview guide [SMIG], which is a series of questions we ask to gain full understanding. This use of SMIG is a manual process. Our tool will include the SMIG and will be able to tell the interviewer, for example, "you have covered these points and still need to address these other ones." Eventually, using this tool, we will produce a draft report that says, based on the findings, "Here are the potential issues." Using this tool will cut down the time spent writing the report, which is most of the elapsed time.

Where else have you seen interest in SMART?

Smith: In fact, there is interest now from inside and outside the Department of Defense. For example, personnel at a large government contractor with good software expertise want to get involved with SOA because many clients ask them to. They are interested in the SEI as an objective broker about SOA approaches and are looking at SMART. We've also talked with other medium to large organizations in this country and Europe.

The material presented in this tutorial is now available in a publicly available two-day course called Migrating Legacy Systems to SOA Environments; for details and registration information, go to http://www.sei.cmu.edu/training/p59b.cfm.

* As of June 2010, SMART stands for *SOA Migration, Adoption, and Reuse Technique*.

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

Software Product Lines FAQs Part VI: Using Software Product Lines with Other Approaches

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: May 1, 2007

Related Links

Training

Software Product Lines - eLearning

Software Product Lines

See more related courses »

Version 5.0 of the SEI's *Framework for Software Product Line Practice*, a conceptual framework that describes the essential activities and 29 practice areas necessary for successful software product lines, was released this summer. In the five years since the release of the previous version, the community using the Framework has grown, and the number of organizations achieving business benefits from a product line approach has grown. At the same time, a number of other emerging technologies have influenced approaches to software product lines. This gave us the opportunity to answer more questions about software product lines and expand this component of the Framework. Following are some of the common questions from Version 5.0.

"Can I use open source software packages in my product line?"

You can if all of the following conditions exist:

- You can live with not having any control over the release schedule of the open source package.
- The provided variation mechanisms of the open source package are appropriate for your need to integrate the package into your product line. Most open source packages have variation mechanisms built in to support the disparate needs of a broad user community. If the adaptations you have to make to integrate the package into your product line can be handled by those provided mechanisms, you'll be in good shape when you have to integrate a new release of the package. If you have to make more substantial changes in the package, be prepared to redo

all of those changes with every new release. Resist the temptation to clone and own. Otherwise, you'll lose the benefits of having a user community.

- The maturity of the package is appropriate for your needs. There are very robust packages available but also an enormous number of alpha or beta releases. Some of those products are also abandoned and never reach a mature state. When selecting a package, do some research about the popularity of that package and find out whether an active user community exists. If it does, you most likely will get a package at least as robust as what you would expect from your own in-house development group.

Also be aware that some open source licenses are crafted in such a way that would require you to make your products, when based on open source packages, also open source. Read the license agreements very carefully.

"Does the software product line approach work in a globalization environment?"

Globalization has two meanings. The first, also called *internationalization* or *localization*, refers to making a software product or software-intensive system work correctly around the world. You can easily see that the software product line approach applies straightforwardly in this case. Different locale requirements correspond directly to product line variation points, and, in fact, a scoping exercise and a product line perspective on requirements can greatly help in identifying those variations. Identifying variations can result in each member of the family being as lean as possible and perhaps letting developers concentrate more on global features as well as locale-specific features.

The second meaning of globalization involves separate development groups located around the world cooperating productively and correctly to build software. In this context, the question is whether product lines can be developed globally. They can. It is necessary to lay some groundwork before effective distributed development can occur, and product lines are not immune from this need. For example, it is very useful to first establish such things as a common development environment, a common configuration management system, and a minimum set of common processes (or at least crisp process interface points), so the exchange of information and artifacts can flow smoothly across site boundaries. Just as in global development for single systems, architecture plays a central role. It crisply defines the work assignments and responsibility boundaries in the system but also among the development parties. In a product line situation, the responsibilities will include the design and development of variation mechanisms for core assets.

"Can a product line approach be compatible with agile development methods?"

The short answer is yes, as demonstrated by the successful use of eXtreme Programming (XP) in Salion's product line effort [Clements 02]. However, the larger point is that the applicability of agile methods is more strongly determined by whether a project's characteristics align with a method's home ground.

Boehm and Turner advocate a pragmatic, risk-driven approach to choosing appropriate aspects from both plan-driven and agile methods [Boehm 04]. For projects whose characteristics stray from agility's home ground, it may still be possible to partition off portions where agile methods can flourish.

One challenge to agile methods' applicability is the principle of simple design, de-emphasizing the importance of software architecture. Within XP, this concept is known as "You Aren't Going to Need It" (YAGNI). As Boehm says, YAGNI works fine when future requirements are largely unpredictable but can be highly inefficient where there is a reasonable understanding of future needs. Because a product line approach inherently means you set out to understand future needs, and indeed base your business strategy on that understanding, you *are* going to need a sufficiently defined product line architecture. However, once it's developed, the software architecture contributes very well to a team's tacit knowledge and can serve as a basis for other agile practices (e.g., for development activities within a partitioned area of the architecture).

"Is a model-driven development (MDD) approach compatible with the software product line strategy?"

Yes. The software product line strategy is compatible with a variety of technical approaches loosely grouped under the model-driven category including MDD, model-driven architecture, and model-driven engineering. The software engineering practice areas in the framework include the activities of a traditional development process but do not constrain how those activities are implemented. The model-driven techniques emphasize the disciplined use of a variety of models, whose currency is maintained throughout the product life cycle. A model-driven approach to software products lines would emphasize automatic product derivation [McGregor 05]. Models would be constructed that are capable of expressing the commonality and variations inherent in the product line's scope. A new product's requirements would be translated into product-specific configurations of the product independent models that are maintained as core assets. Tool-supported translation, laid out in a production plan, would be used to derive products. In fact, model-driven development could be thought of as a production strategy, as described in Section 2.1 of the framework.

"Does a system-of-systems (SoS) context rule out the use of software product lines?"

No. On the contrary, software product lines can help reduce the complexity of an SoS context. An SoS comprises independent, self-contained systems that, when taken as a whole, satisfy a specified need. Many software-intensive contexts today are systems of systems. The complexity involved can be daunting. However, in many cases, there is considerable commonality among some of the self-contained systems. Suppose, for example, that the SoS involves 200 separate systems that all must interoperate. But suppose further (as is often the case) that there are some clusters within those 200 that have considerable commonality and whose variations could be handled economically. By making those clusters into software product lines, you can tame the interoperability issue into a smaller, more manageable set of interfaces and thereby reduce the complexity of the SoS. Moreover, the economic advantages and predictability associated with software product lines will be highly beneficial to the SoS in question.

"Is there any connection between service-oriented architectures (SOAs) and software product lines?"

SOAs and software product line approaches to software development share a common goal. They both encourage an organization to reuse existing assets and capabilities rather than repeatedly redeveloping them for new systems to achieve desired benefits such as productivity gains, decreased development costs, improved time to market, higher reliability, and competitive advantage.

"How are SOA and software product line approaches alike?"

Both approaches promote reuse by developing applications or products based on a set of reusable components. Those components are developed with well-defined interfaces and processes that specify how the components are to be used, which enables applications or products to be produced in less time.

Adopting either approach requires implementing similar organizational policies and practices necessary to adopt a new technology or a new way of doing business. SOA and software product lines share many of the same organizational issues, such as planning, funding, tool support, training, and the need to change the organizational mindset toward reuse. When starting to use either approach, it is imperative for both SOA and software product line efforts to have organizational commitment and a champion.

Both approaches focus on identifying the application building blocks or reusable components associated with the application(s). In SOA, services represent the reusable building blocks. Core assets are the basis for production of products in a software product line. Separate teams may be employed to develop the reusable components and the applications. Small or pilot studies help develop skills to evolve the organization toward an SOA or software product line capability.

In both cases, the initial building blocks may come from legacy systems. Identifying and retrieving product line assets or services from existing systems in order to obtain the benefits of reuse are equally difficult. Documentation and tool support aid this effort.

Application or product development for both approaches is orchestrated in a similar manner. Inputs include the requirements for a particular application or product, reusable components, and the details of how these components are to be used to build the application or product.

"How are SOA and software product line approaches different?"

While the goals and the use of reusable components in the SOA and software product line approaches are similar, the processes by which the two compose systems are very different.

A software product line is, fundamentally, a set of related products. Each product is formed by taking applicable components from the base of common assets, tailoring them as necessary through preplanned variation mechanisms such as parameterization or inheritance, adding any new components that may be necessary, and assembling the collection according to the rules of a common, product-line-wide architecture under the auspices of a production plan. New or updated core assets are rolled back into the core asset base for future systems.

In SOA, it is not necessary for the reusable component(s) to come from a centralized, organization-controlled service base. Multiple organizations may provide the services leading to the possibility that services may change or disappear without notification. While multiple organizations can have responsibility for the core asset base in the software product line paradigm, that is not the usual case.

SOA addresses the issue of variation through orchestration (coordinating the participating services), service versioning, or extensible XML data types (a process of evolving from one format to another without requiring central control of the format).

The product line architecture is a software architecture that satisfies the needs of the products within the product line's scope. It is a key core asset. SOA is not the software architecture of the system; it is a design philosophy and an approach to software development where

- Services provide reusable functionality with well-defined interfaces.
- An SOA infrastructure enables discovery, composition, and invocation of services.
- Applications are built using functionality from available services.

The product line architecture establishes the quality goals for a system—its performance, reliability, modifiability, and so forth. Since SOA implementations may span enterprise boundaries, the quality attributes are dependent on the Quality of Service (QoS) of each of the included services. Desired end-to-end qualities may be achieved in specifically engineered applications. However, if the services are used in a different context, they may not meet the expected QoS. Service developers need to understand the functional and QoS requirements of potential service users.

In a software product line, an established process for updates to the core asset base is followed as the product line evolves, as more resources become available, as fielded products are maintained, and as technological changes or market shifts affect the product line's scope. Unlike SOA, the core assets in a software product line approach include non-software assets as well as software components. Product line requirements, domain models, test cases, and other assets provide significant strategic advantage. Also included in the core asset base is a production plan prescribing how the products are produced from the core assets. SOA employs an SOA governance to facilitate planned reuse of services. SOA governance means the creation, deployment, enforcement, and verification of policies throughout the entire life cycle of SOA artifacts. SOA governance platforms may provide easy service discovery through a centralized service registry and management tools for planned reuse (on the service level), such as tracking subscribers to services, negotiating service level agreements

(SLAs), communicating change requests and actual changes in service interfaces and data types.

"Can SOA and software product lines be used together?"

It is possible to build a stand-alone (i.e., non-product-line) application using SOA and to build a software product line without using SOA. In that sense, the two approaches are independent. However, it is also possible to combine the two approaches. In particular, it is possible and feasible for an organization to use services as reusable core assets with which to build products in a software product line. That is a focus of the Using Externally Available Software practice area. Also, service providers and SOA application developers could take a product line approach to the development of services."

Can the framework provide guidance for a move to SOA"

Organizations moving to an SOA approach can benefit from software product line information captured in other practice areas as well. The practice areas related to organizational management could help organizations understand important issues in adopting a new reuse-based technology. And those related to technical management and software engineering could help organizations analyze legacy systems, determine make/buy/mine/commission decisions, use existing available software, and understand risk. Documents similar to the product line adoption plan (describing the desired state of the organization and a strategy for achieving that state) and the product line production plan (which prescribes how the products are produced from the reusable components) provide excellent templates for organizations moving to planned, reuse-oriented environments.

Planned and systematic reuse, exploiting economies of scope, and other important software product line issues could feed into the SOA design philosophy to help manage the growth of services and application developments, understand the dependencies between services, and determine the impact of service changes to the applications. The software product line approach would help control complexities created by the combinatorics of services and applications that occur over the entire life cycle of a system.

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

[Clements 02]
Clements, P. & Northrop, L. *Salion, Inc: A Software Product Line Case Study* (CMU/SEI-2002-TR-038, ADA412311). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002.

[McGregor 05]
McGregor, J. *Preparing for Automated Derivation of Products in a Software Product Line* (CMU/SEI-2005-TR-017, ADA448223). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2005.

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

Being Your Own Boss—Part II: The Autocratic Manager

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: May 1, 2007

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

This is the second in a series of columns on being your own boss. The prior column discussed ideal jobs, what developers like about their work, and what they don't like. It also described how managers' and developers' views differ about project success and what this means to developers, managers, and their organizations. In this column, I talk about autocratic managers, autocratic behavior and why it is common, and the consequences of autocratic behavior. In subsequent columns, I will discuss why an autocratic management style is not appropriate for modern development work, the alternatives to autocratic management styles, and how each developer can take charge of his or her personal working environment to turn a seemingly unpleasant job into an ideal one.

The Autocratic Boss

Autocratic bosses are common. Starting with the construction of the early pyramids in Egypt, the literature is full of examples. Bosses have historically been slave drivers, demons with whips, or guards with guns. Historically, workers may have been slaves or prisoners, but that is not true of typical working environments today. However, in development work, bosses typically have very substantial power, and workers do not. This power confers a level of authority that bosses can use in autocratic ways.

Even when the boss is charming and seemingly respectful and friendly, that boss would still be an autocrat if he or she behaved unilaterally. Autocrats do not really consider the feelings or views of their workers. While this may sound extreme, it really is not. The true test is whether the boss actually considers your needs and views in making

decisions. Merely pretending to listen to your opinions does not change the facts. A typical autocrat in effect says, “Let’s compromise and do it my way.”

Why Is Autocratic Behavior Common?

Autocratic behavior is common because it can have substantial advantages for the autocrat. For example, autocratic decisions can be made quickly. Assuming that the autocrat is technically competent, this could be advantageous in dangerous situations like military campaigns or natural disasters. Autocratic behavior could also be effective when the autocrat-boss was better informed and more competent than the workers. This is often the case for simple and highly repetitive work. The autocrat can also get substantial personal rewards from unilateral action. Every time some autocratic command is promptly and successfully carried out, the autocrat’s belief in his or her competence and infallibility is reinforced. This self-reinforcing characteristic of power led Lord Acton to say over 100 years ago that “power corrupts, and absolute power corrupts absolutely.”

While this conclusion has been widely recognized and quoted, there is another similar conclusion that is not as well known: “Petty powers are most corrupting.” Philip Zimbardo found this to be the case in simulated-prison experiments at Stanford University in the 1970s [Gladwell 1973]. Zimbardo enlisted volunteers to act like prisoners and guards in a simulated prison. They had built a facility just like a prison in the laboratory basement and actually locked up the volunteer “prisoners.”

What Zimbardo found was that, after only a couple of days, the “guards” started to behave so abusively that one of the “prisoners” actually had a severe emotional breakdown. He had to stop the planned two-week experiment in only six days. He concluded that ordinary people will often act in authoritarian ways when they are given even menial jobs that have some minor but absolute powers.

Zimbardo’s conclusion can be seen in the behavior of many bureaucrats: they often tend to use their limited powers in arbitrary and highly authoritarian ways. This is what makes some clerks, guards, or other support people insist on strictly interpreting the rules although that interpretation would make no logical sense in the specific case at hand. This kind of strict adherence to work rules makes organizations unresponsive and hard to work with. It can also be expensive and demotivating.

Why People Are Autocratic

There are four reasons for people to behave autocratically.

1. There is a crisis.
2. There is a power vacuum, and they feel they must take charge.
3. They act autocratically because that is the way such jobs have always been done.
4. They get emotional benefits from being autocratic.

The Crisis—In times of crisis, rapid decisions are often required, and a single authority is usually thought to be most effective. Most people understand and accept that this is the best way to behave in such cases.

The Power Vacuum—A power vacuum can occur in a crisis when the leader is either killed or otherwise unavailable and someone takes over. Such cases can arise in almost any situation, and it may be seen in combat when the commanding officer is killed, and a sergeant or even a private takes charge. While this situation need not lead to autocratic behavior, it generally does when nobody else seems willing to or able to make the needed decisions.

Force of Habit—While power vacuums can occur in development, the force-of-habit case is more typical. Development managers have generally managed in this way so pretty much everyone does. Furthermore, since it seems so natural and expected for the boss to make all of the decisions, force of habit tends to create power vacuums. Nobody but the designated boss feels able to make decisions.

Emotional Reinforcement—Emotional reinforcement presents an entirely different situation. When the person has clear authority and resists either suggestions or

appeals to common sense, it is generally a good idea to keep quiet and do what you are told. If this person is your boss, it may not be clear whether he or she would accept suggestions or not. This is when you may have to conduct tentative tests to see how your suggestions are received. While I have only seen a couple of truly autocratic managers in 50+ years of development experience, they do exist, and they can be both unpleasant and threatening to work for, particularly if, like me, you like to make your own decisions.

The Consequences of Autocratic Behavior

While autocratic environments are not pleasant places to work, they can be reasonably efficient when the boss is both competent and fully capable of directing the work. Even in these cases, however, it has long been known that autocratic management styles demotivate the workers and produce less than optimum workplace performance.

One reason for this is explained by James Surowiecki in his book *The Wisdom of Crowds* [Surowiecki 2004]. He cites many examples of how groups typically make much better decisions than even expert individuals. This is particularly important for development groups and is the reason that autocratic behavior is ineffective and often even counterproductive. This is true regardless of how pleasant and friendly the autocrat is; the problem is unilateral behavior.

An Example

Since it is often hard to recognize that you are working in an autocratic environment, an example may help. In one case, a development team was starting a new project and management told the team leader that the job had to be completed in nine months. The team leader then produced an overall plan with key milestones for

- design-specification sign-off
- detailed design complete
- code complete
- functional testing
- system testing
- customer-acceptance testing

He then met with the entire development team and reviewed his proposed plan. Over the next couple of hours, he answered all of the developers' questions and made some minor additions and adjustments to the plan. In the end, even though none of the developers felt that the nine-month schedule could be met, the team agreed to the plan pretty much as the team leader had originally proposed it. This team leader then told management that his team now had a plan to deliver the finished product in the desired nine months.

The question is: "Is this autocratic behavior?" Most developers would say no. This is how most of their projects have always been run. They believe that they could have spoken up and made changes to the plan if they had wanted to. Furthermore, most of them also felt that the team leader knew more about planning than they did, and they may even have believed that he or she had produced a better plan than they could have. Finally, since they had to meet management's nine-month schedule and this plan did, they didn't have anything better to suggest. The team leader would also likely argue that he or she was not being autocratic. After all, there was a full team review of the plan and all of the team's suggestions and changes were incorporated.

While it is true that this style is nothing like that of the despot or slave driver of old, it still results in a unilateral plan that was produced with little or no team input. While minor changes were made, the plan had the original resources management allocated, it produced the product that they had specified, and it met management's suggested end date. Such plans commonly have three important characteristics.

1. They do not guide the developers in doing the job.
2. They do not have the full commitment of the team members.
3. They are rarely met.

The next question, of course is: “If this is autocratic management, how else could you produce a plan?” The answer is to truly involve the team in making its own plan. However, there are typically two objections to doing this.

1. It would take too long.
2. The developers wouldn’t know how to make a plan.

Of course, if you don’t mind getting inaccurate and largely useless plans, it doesn’t make much difference how you produce them. However, if you want accurate and useful plans, it might make sense to consider alternatives. I address these points in subsequent columns.

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 Dan Burton, David Carrington, Julia Mullaney, Bill Nichols, Bill Peterson, and Alan Willett.

In Closing, an Invitation to Readers

In these columns, I discuss development issues and how they impact the work of engineers 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. Better yet, include a story or brief anecdote to illustrate 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

References

[Gladwell 1973]
Malcolm Gladwell discusses Zimbardo's work on pages 152 to 155 of his book *The Tipping Point: How Little Things Can Make a Big Difference*, Little, Brown and Co., New York, 2000. The original reference is Henry, C.; Banks, W.C.; and Zimbardo, P.G. (1973), “A study of prisoners and guards in a simulated prison.” *Naval Research Review*, 30, 4-17.

[Surowiecki 2004]
Surowiecki, James. *The Wisdom of Crowds: Why the Many Are Smarter than the Few and How Collective Wisdom Shapes Business, Economies, Societies, and Nations*, Doubleday, New York, 2004.

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.

Share This Page

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

ArchE-the Architecture Expert

NEWS AT SEI

Author

Len Bass

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

Software Architecture

This article was originally published in News at SEI on: May 1, 2007

The Architecture Expert (ArchE) tool is an experimental assistant to the architect that the SEI has been developing for several years. With the current release of Version 2.1, it is available for non-commercial use.

One of the SEI's axioms is that quality-attribute requirements such as those for security, performance, modifiability, usability, and so forth have a dominant influence on a software architecture. A reasonable question in light of this axiom is "How can quality-attribute knowledge be codified to help the architect during the design process? " ArchE is a system that embodies our efforts to codify quality-attribute knowledge.

ArchE is based on four different concepts:

1. Quality-attribute scenarios. ArchE requires that quality-attribute scenarios be specified in a six-part formal structure involving a stimulus, a stimulus source, an environment, an artifact being stimulated, a response, and a response measure.
2. Responsibilities. Activities within the system being designed are represented by responsibilities. As the design process progresses, responsibilities are added, split, or modified to support the achievement of particular quality-attribute requirements.
3. Reasoning frameworks. Quality-attribute knowledge is encapsulated into reasoning frameworks. A reasoning framework includes abstracting relevant quality-attribute knowledge from an architecture, evaluating whether the architecture meets the requirements for that quality attribute, and proposing tactics as improvement mechanisms.

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 »

4. Architectural tactics. An architectural tactic is a means of satisfying a quality-attribute response by manipulating some aspect of the architecture.

One analogy for ArchE is tax preparation software. That is, tax preparation software encapsulates a great deal of knowledge about the structure of the tax code and has the ability to check for consistency, but the information on which the tax return is based and the correctness of the resulting return depend on the information provided by the user. ArchE has knowledge of quality attributes and how to relate quality requirements to architecture design, but has no knowledge of the semantics of the system being designed. The architect must provide ArchE with information about system semantics.

To use ArchE during the design process, the architect provides design knowledge and judgment, and ArchE provides quality-attribute knowledge. The dialogue between the architect and ArchE proceeds as follows:

- The architect inputs
  - features that need to be computed by the system being designed
  - quality-attribute requirements specified as scenarios
- ArchE asks for information necessary to determine quality-attribute behavior, such as execution time for various features or cost of change of various features.
- ArchE
  - proposes a design;
  - identifies quality-attribute scenarios that are not achieved by that design;
  - presents a list of tactics that might improve the design; for example:
    - Insert an intermediary to improve the modifiability in a certain area.
    - Introduce concurrency in an area to improve performance.
- The architect chooses one or more tactics, and ArchE applies these tactics to produce a new proposed design.
- The architect then may need to provide additional information to the new design such as a meaningful name for a common service or computation time for an intermediary.
- This process repeats until
  - the quality attribute requirements are met;
  - the architect is happy with the result; or
  - ArchE has no more proposals.

Version 2.1 of ArchE is now available online. It includes quality-attribute knowledge about modifiability and real-time performance. It has been used successfully in a graduate-level software architecture class at Clemson University. ArchE depends on the following software, which is all freely available for non-commercial use:

- JESS. JESS is a rule engine written in Java that has the ability to interact with pure Java code. JESS is used to manage the rules that ArchE uses for the design process and for interacting with the reasoning frameworks.
- MySQL. MySQL is a database-management system. It is used to store portions of the JESS fact base that persist across different design alternatives. It is also used to interact with external tools that must either access the output of ArchE or provide input into ArchE.
- Eclipse. Eclipse is an open-source development platform. ArchE uses Eclipse to manage its user interface.

The target audience for this release includes educators who want to incorporate software architecture design concepts into their classes and others who are interested in exploring the use of tactics within an architecture-design process.

If you are interested in using ArchE on a commercial project, contact Greg Such at gsuch@sei.cmu.edu.

About the Author

Len Bass is a senior member of the technical staff at the Software Engineering Institute who participates in the High Dependability Computing Program. He has written two award-winning books on 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.

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.

OCTAVE Allegro Speeds Up the Risk Assessment Process

NEWS AT SEI

Author

Pamela D. Curtis

This article was originally published in News at SEI on: May 1, 2007

Less than a century ago, the ratio of tangible assets to intangible assets in the U.S. economy was 70 to 30. In the new economy, that ratio has been inverted, observes New York University professor Baruch Lev: "In the past several decades, there has been a dramatic shift, a transformation, in what economists call the production functions of companies—the major assets that create value and growth. Intangibles are fast becoming substitutes for physical assets" [1].

Among the intangibles are *information assets*, such as intellectual property, patient records, and customer data. Many organizations have not realized the value of these assets until a security breach has compromised them and resulted in substantial loss.

The SEI has introduced a risk-assessment method, OCTAVE Allegro, that can help businesses identify their information assets and determine how those assets are at risk. Allegro is described in the SEI report *Introducing OCTAVE Allegro: Improving the Information Security Risk Assessment Process*.

Allegro is a variant of the SEI's OCTAVE (Operationally Critical Threat, Asset, and Vulnerability Evaluation) method, which since 1999 has been used by many organizations to identify critical assets and risks. Another variant of OCTAVE, OCTAVE-S, is designed for organizations of about 100 or fewer employees. Allegro is not intended to supplant these previous OCTAVE methods; it is an alternative that provides a streamlined process focused on information assets.

Like the previous methods, OCTAVE Allegro can be performed in a workshop-style, collaborative setting and is supported with guidance, worksheets, and questionnaires, which are included in the appendices of the Allegro report. However, OCTAVE Allegro is also well suited for use by individuals who want to perform risk assessment without extensive organizational involvement, expertise, or input.

(< 5 minute) survey.

Origins of the Method

Government agencies of Clark County, Nevada, adopted OCTAVE as a way to comply with federal Health Information Portability and Accountability Act (HIPAA) requirements to protect the privacy of personal information collected through their social services. They had conducted two OCTAVE pilots and were looking for a way to easily institutionalize OCTAVE risk-assessment principles and practices at the operational-unit level. SEI staff members developed Allegro in response to that need.

"We reduced the amount of risk-assessment knowledge needed, simplified instructions, and provided worksheets where the risk view can be summarized in one place," OCTAVE Allegro developer Rich Caralli explains. "At the end of a five-hour workshop, the participants already had half of a risk assessment done. We named the new method 'Allegro' after the musical term meaning 'in a quick and lively tempo' because we think it will save time and energy." Clark County went on to do train-the-trainer classes and is now implementing Allegro organization-wide at the operational-unit level.

No risk analysis or IT background is required to use Allegro. Allegro is typically implemented by business-unit managers working with their staffs because together they are the people who often know best what the information assets are.

"Exposure to Allegro gives people an opportunity that they haven't had before to think about risk, to combine their knowledge of their businesses with a new understanding of risk," says Caralli. "We've had a few situations where people have excused themselves temporarily from an Allegro training class because they have suddenly become aware that an asset under their control was at risk and have realized they had to do something about it immediately."

Information Assets and Containers

The primary focus of the OCTAVE Allegro method is the information asset. All other assets important to the organization are identified and assessed in the context of the information assets to which they are connected. This eliminates potential confusion about scope and reduces the possibility that extensive data gathering and analysis will be performed for assets that are later found to be poorly defined, outside of the scope of the assessment, or in need of further decomposition. "If you don't identify the asset accurately early, you carry that mistake deep into the risk-assessment process," says Caralli. "By focusing on information-asset identification, you avoid that."

For example, at a high-level asset view, a certain system might be identified as being critical. But closer examination reveals that it's actually particular data on the system that is critical. The system is just one of potentially many places where that data is stored, transported, and processed, both inside and outside the organization. In Allegro, these places are referred to as *containers*. They are usually some type of technical asset—hardware, software, or system—but can also be a physical object such as paper or even a person. An information asset's containers can become points of vulnerability where it is at risk. So an important part of the Allegro method is identifying each information asset's containers.

Components of the Method

The OCTAVE Allegro method consists of eight steps that are organized into four phases, as illustrated in Figure 1. In phase 1, assessment participants develop risk-measurement criteria consistent with organizational drivers—the organization's mission, goals, objectives, and critical success factors. During the second phase, participants create a profile of each critical information asset that establishes clear boundaries for it, identifies its security requirements, and identifies all of its containers. In phase 3, they identify threats to each information asset in the context of its containers. In the final phase, participants identify and analyze risks to information assets and begin the development of mitigation approaches.

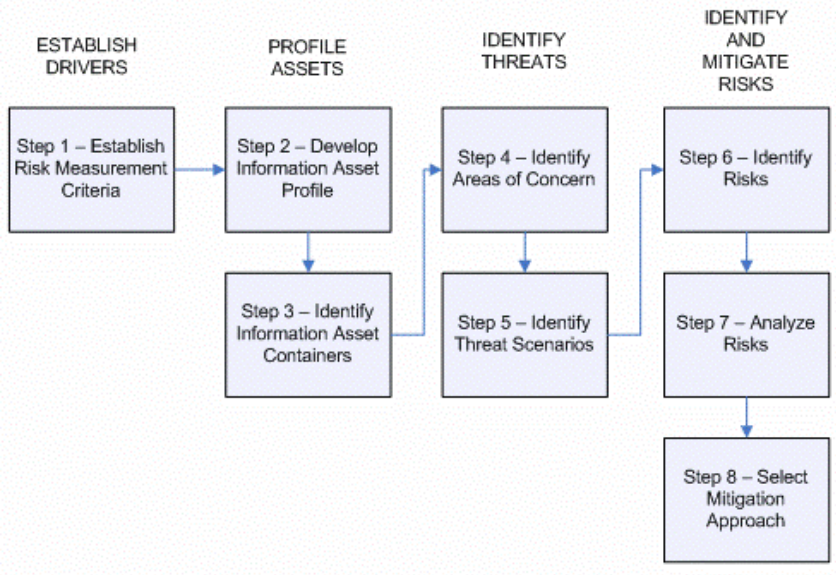

Figure 1: OCTAVE Allegro Roadmap

How To Implement Allegro

There are a few options available for using Allegro:

- Download the Allegro technical report and use it as a guide. All of the worksheets and questionnaires used in the method are included in the report.
- Take the OCTAVE training offered by the SEI and SEI partners, which includes training in Allegro. The training covers guidance for implementing and institutionalizing Allegro.
- Arrange a custom engagement with the SEI to receive training on site (including train-the-trainer classes), mentoring through the implementation process, and help with institutionalization of Allegro.

References

[1]
Webber, Alan M. "New Math for a New Economy." *Fast Company 31* (December 1999): 214.

The Double Challenge in Engineering Complex Systems of Systems

NEWS AT SEI

Authors

Bill Anderson

Philip J. Boxer

Edwin J. Morris

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: May 1, 2007

Traditional software engineering practices were defined when development was largely controlled by organizations that could set relatively stable requirements, build to those requirements, and deliver a system to the customer. More recently, increasingly complex and dynamic customer demands have focused attention on coordinating activities of multiple organizations and systems within an enterprise to perform a number of tasks or deliver tailored responses. This change in focus from a specific delivered system to the need for flexible capabilities is reflected in product lines, families of systems, and other recent advances in software engineering practices.

However, to meet customer expectations with the emerging, complex systems of systems required to support integrated military strategies, homeland security responses, and nationwide health information networks, system developers must meet a double challenge (see Figure 1):

1. A **governance challenge** of collaborating with an increasing number and diversity of enterprises.
   This challenge includes developing approaches that support cooperation across unrelated enterprises with no unifying controlling structure.

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.

2. An **agility challenge** of providing situation-appropriate responses in changing situations.
   This involves selecting technologies, processes, and structures that are sufficiently agile to support the desired response.

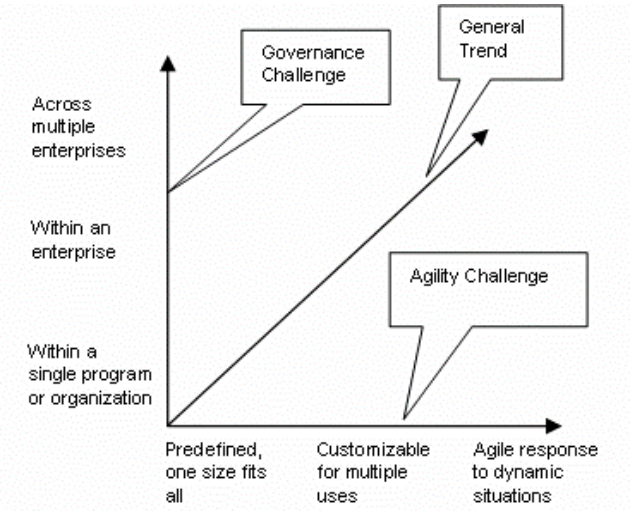

Figure 1: The Double Challenge

The Governance Challenge

The governance challenge involves the changing nature of the collaboration needed to build computer systems. For most of the history of computing, collaboration was needed only to coordinate activities involving point-to-point interfacing of systems within a well-defined program. As anyone experienced with building or maintaining relatively complex software can attest, even this level of collaboration required coordination of the activities of multiple personnel, perhaps with competing interests. However, there was normally one authority that could make and enforce decisions.

In more complex cases, collaboration was needed to coordinate the activities of multiple systems within a single enterprise. This situation was typical of an organization that attempted to relate and integrate the activities of several systems. The need to integrate multiple systems became the driving force behind efforts to integrate data and processes in an enterprise-resource-planning (ERP) system or to construct a common situational picture of the location of friendly and enemy assets by fusing the data contained in multiple military systems operated by the Department of Defense.

Of course, enterprises of any significant size are not homogeneous; they typically consist of multiple levels of relatively autonomous sub-enterprises (e.g., branches, divisions, directorates, teams). However, a hierarchy is assumed to exist that can resolve conflicts in such a way to produce an internally consistent whole.

But system-integration activities become increasingly complex as they cross organizational boundaries. These activities are commonly managed by creating a special management organization with authority that spans these boundaries. The success of these cross-cutting organizations depends on the degree to which they can establish centralized coordination and control of activities—that is, to establish authority over the activities and systems to be integrated as well as over the integration activity itself.

We are familiar with the difficulties this presents to such cross-cutting organizations. The interests of the cross-cutting organizations often come into conflict with those of individual organizations that make up the whole. In principle, these difficulties can be resolved by appeal to the higher authority of the enterprise as a whole. In practice this can be infeasible.

When the cross-cutting approach is extended to multiple enterprises without a unifying hierarchy, a discontinuity occurs (illustrated along the vertical axis of Figure 1). Moving from bottom to top, collaborators become increasingly autonomous and their motivations, policies, procedures, and capabilities become increasingly diverse. In fact, because of the fluid environment in which complex systems must execute, the owners of those systems cannot fully anticipate who these collaborators will be.

To meet the challenge posed by collaboration with an increasing number and diversity of enterprises, processes and strategies must be developed that support negotiation of relationships with these sometimes unanticipated partners. Therefore, we must develop ways of negotiating *collaborative governance* across unrelated enterprises in rapid order.

The Agility Challenge

The agility challenge, illustrated in Figure 1 along the horizontal axis, describes the way that an enterprise's[1] computing systems can respond to the needs of its customers. In simple cases, an enterprise can define the systems it needs to provide to its customers, put together (or otherwise acquire) these capabilities, get them into the right hands, and keep them there. This is the strategy used by enterprises in building tightly focused applications for general use (e.g., Microsoft Word).

In more complex cases, a single predefined system capability is insufficiently flexible to meet the customer's demands. In such cases, enterprises try to set parameters for their applications, processes, and organizational structures that customize them to multiple customer situations. Technologies and systems with customizable interfaces and assembly strategies, such as product lines or families of systems, are employed to increase the range of customer needs that can be met by the organization. Another example is service-oriented architecture (SOA), which is intended to maximize the extent to which services can be composed.

Solutions to these two cases are driven by what the supplier can provide, either in a predefined system or through a customizable set of applications designed to work within some prescribed set of behaviors. However, customers are expecting ever more flexible responses to rapidly changing situations. In this evolving situation, the specific context in which the customer wants to employ a capability becomes a critical determining factor. As a result, it is no longer possible for the technologies and systems to be under the control of specific individuals or organizations. The customer is adding a new level of composition and synchronization of components.

Thus, the agility challenge represents a second discontinuity—between situations controlled by suppliers and situations controlled by the situational needs and demands of users

Implications of the Double Challenge

Along the vertical (governance) axis, interactions between different organizations become increasingly common. This tendency is evidenced in the engineering community by the widespread push for increasing interoperability and standards that support it. It is at the root of efforts to integrate the diverse systems of commercial enterprises into supply chains, to develop common operational pictures across domestic and allied military forces, and to provide emergence response capabilities crossing military, police, government, health care, and other networks.

Along the horizontal (agility) axis, there is increased recognition in commercial, government, and military sectors that advantage is best gained by developing system capabilities that can be rapidly aligned in new ways to support responses to changing demands. For commercial organizations, this means that supplier relationships must meet changing needs—whether by supporting rapid changes to products and manufacturing approaches or by providing new ways for service delivery as determined by the customer. Ideally, the alignment of organizational structures, processes, and systems capability is determined by the demand.

Thus, the general trend is clearly up and to the right in as depicted in Figure 1. However, given the traditional methods of building systems, the comfort zone for building systems is actually down and to the left as shown in Figure 2.

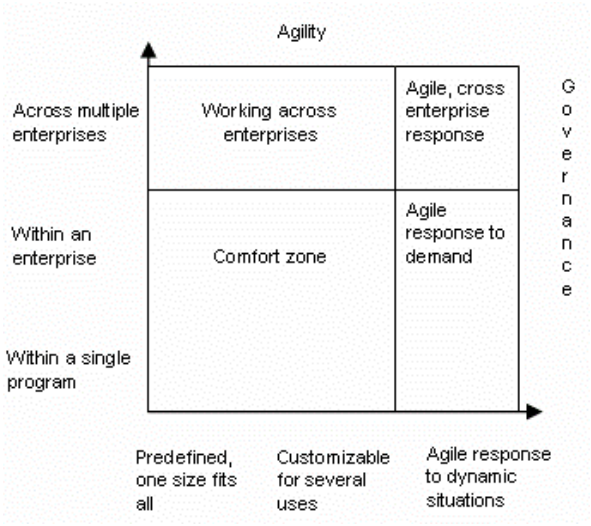

Figure 2: The Comfort and Other Zones

The top right quadrant of this chart involves cross-enterprise and agile response to changing needs and demands of users. This goal is perhaps best developed today in the power-to-the-edge strategy of the U.S. Department of Defense (DoD), which has recognized that only at the edge can it respond appropriately to the demands being placed on it for cross-command (e.g., services, allied militaries, and non-traditional allied) responses.

We believe that the future of software engineering will be dominated by the double challenge of developing governance approaches that can work across enterprises and identifying ways to meet demands for increasing agility in the capabilities that are provided.

We are concerned that focusing on developing engineering strategies to improve or fix individual symptoms of the double challenge, such as poor coordination of development efforts across organizations and problems with configuration management, may be helpful in some situations but ultimately will not solve the problem.

It is more likely that fixes for individual symptoms along one axis of the double challenge will actually complicate problems along the alternate axis. For example, developing a new, virtual organization that imposes a hierarchy across enterprise boundaries may lead to reduced flexibility in response to new user demands. The key question is how enterprises can develop the ability to work across enterprise boundaries while simultaneously providing the agility to respond to the changing demands of the customer.

A starting point involves distinguishing different types of systems of systems based on the type of authority possible and the ability of that authority to control behavior. This approach allows us to begin to characterize requisite engineering practices. Of particular interest to us are those practices that support distributed collaboration[2] (e.g., power-to-the-edge). A critical activity for the future is to consider the double challenge (i.e., governance and agility) in relation to achieving distributed collaboration.

In response to the double challenge, the SEI is developing the System-of-Systems Navigator[SM], an integrated set of principles, tools, models, techniques, and improvement cycle activities. The SEI is currently developing capabilities to recognize different types of systems of systems along the dimensions outlined above. Of particular interest are those practices that support distributed collaboration (e.g., power-to-the-edge) risk assessment in this complex space through modeling and gap analysis between the required collaborative constituent components.

For more information on the System-of-Systems Navigator, contact Suzanne Garcia (smg@sei.cmu.edu).

1 We have used *enterprise* here to mean any sort of entity that must respond to the second challenge. That enterprise may entail one or more organizations.

2 As opposed to directed collaboration that has a controlling authority

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, 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 (ISIS) 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.

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.

Survivability Challenges for Systems of Systems

NEWS AT SEI

Authors

Robert J. Ellison

Carol Woody

This article was originally published in News at SEI on: June 1, 2007

The Changing Operational Environment

As systems become loosely coupled groups of software modules functioning independently and assembled dynamically to exchange information and perform shared services, the ability to establish control boundaries for system and component certification and accreditation grows increasingly difficult. The same set of interconnected software modules could be implemented within a single operating system or distributed internationally. Also, connectivity could be peer-to-peer, wireless, or any number of combinations. These varying implementation choices represent drastically different threat environments. Software, however, is not currently built to consider this range of variability.

Increasing system complexity, required system adaptability, and new operational mission needs are driving software and system developers to the expanded use of technologies such as Web services and design approaches such as service-oriented architectures. The new technologies and changing operational environment raise software risks that are not addressed by existing operational risk approaches. Newly developed components join an existing operational environment and must be connected with older technology for operational effectiveness. Too often we see a patchwork of software components stitched together with home-grown solutions. It might work, but with very little or no assurance. It is likely inflexible and potentially unpredictable.

Operational support and sustainment efforts must consider that the operational environment will grow increasingly fragile as the need for increased flexibility and solutions to meet immediate needs drive development to field solutions based on newer, less tried tools and techniques that must integrate seamlessly with legacy systems. Changes in the operational environment are also occurring, independent of

development activities. Continuous hardware and operating system upgrades take place as support for older versions expires. Vulnerability monitoring and incident mitigation introduce changes to infrastructure configurations and components such as firewalls and routers. Growing complexity and increasing interdependency of operational components can overwhelm problem analysis and response mechanisms. As new development joins the implemented environment, limitations on consideration of operational needs will accentuate these challenges.

Risks

- The operational situation may change before problem analysis is completed.
- Rapidly changing threats and attack patterns mean that quality estimates of the probabilities may not be available when the threat must be addressed.
- Interdependencies and complexity can hide risks that may not be observed until deployment. It may be difficult to identify risks for new usage of an existing system or when systems with unanticipated configurations are linked.
- With distributed processing, we have to protect data at rest as well as in transit, but actual transfers and resting points may not be known.

Impacts

- The effects of a component change or failure are difficult to contain, and external interdependencies may be unknown. An insignificant local change or error can be magnified by unexpected or poorly understood interdependencies.
- Simplifying analysis by considering only critical components is less effective. It may be hard to find components that are not critical in some context.
- The difficulty of predicting the behavior of a system of systems (SoS) is compounded by multiple control-points that do not share common risk mitigation strategies.

Mitigation Concerns

- Operations are increasingly non-stop. A change in response to an operational state cannot require system restarts and may have to be done in minutes or hours rather than weeks. It is impractical to require manual changes at each application or to require redeployment for configuration and security changes.
- The rapid evolution of hardware and technology implies that a homogeneous deployment eventually becomes heterogeneous. The heterogeneity creates the potential for multiple and potentially conflicting approaches to security and the risk of incomplete solutions.
- Accountability and traceability could be difficult to incorporate in a service-based security infrastructure and will be especially difficult to provide in a distributed and heterogeneous environment.
- The sheer number and diversity of software and hardware components limits the ability to synchronize upgrades and deployments. Such synchronization is even less likely across multiple organizations. Interoperability with legacy components or those with unapplied upgrades is a given.

Increased Importance of Operational Survivability

The survivability of an operational process (mission thread) depends on the availability of a specific set of functions. In the systems-of-systems environment, those functions can span multiple systems [Maier 96]. Those systems must support multiple mission threads and provide functionality for local processes (local threads), which the stand-alone system was originally designed to address.

Survivability focuses on what can go wrong (the risks), the consequences of those risks on the operational process (mission thread), and the possible mitigation options for those risks. Survivability emphasizes continued operations in the presence of error and recovery following a failure. A response to a failure is not necessarily a system response but may be a change in procedures or an acceptable modification of the mission.

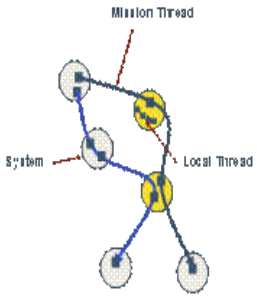

Figure 1. Mission threads cross system boundaries

Survivability must account for the complexity of a networked operations environment, which can be distributed globally. Figure 1 presents a simple view of the global infrastructure environment. Figure 2 captures the variances among levels of connectivity, which can be described as strategic, operational, and tactical (see description in Table 1). Administrative controls are distributed across multiple systems, and there is limited visibility for system behavior beyond the boundaries of each local administration. Global guidance of systems has a role at the strategic level, but at the tactical level, systems must be able to act autonomously because communications and any associated controls are not as reliable.

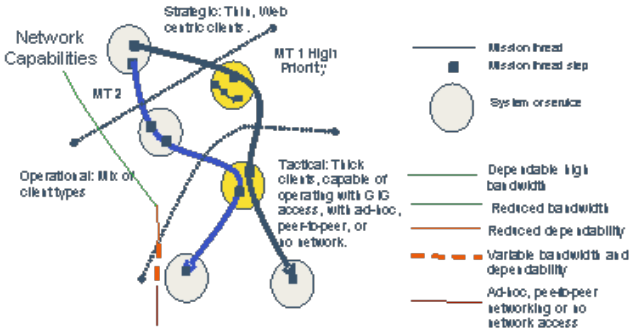

Figure 2. Infrastructure capabilities vary across the organization

Survivability solutions cannot assume homogenous configurations. In reality, the operational capabilities (including bandwidth and latency) will vary across the strategic, operational, and tactical infrastructure levels of the organizational infrastructure. The risk-mitigation tactics must also allow for variances in software architecture induced by these differences. Existing systems and components are expected to evolve to greater interoperability and dependency on shared infrastructure capabilities, reducing the level of functional independence and increasing the potential for reliability and dependability concerns.

Table 1. Variations in application architectures

Context	Network Dependability	Clients
Strategic	Relatively dependable network	Web-centric an option: i.e., thin clients with applications executing on the server
	Noticeable variations in	Modes: May need to shift from thin to

Operational	bandwidth and reliability	thick client if operating conditions deteriorate
Tactical	May have to resort to ad-hoc and peer-to-peer configurations	Thick clients: Some applications must run locally with limited or no network access

Mission-thread requirements crossing system boundaries (joint among systems) will compete for resources with system-provided functions. SoS mission-thread execution will compete with local threads and other multi-system threads assigned to the same system for priority, bandwidth, and other resources (as shown in Figure 3).

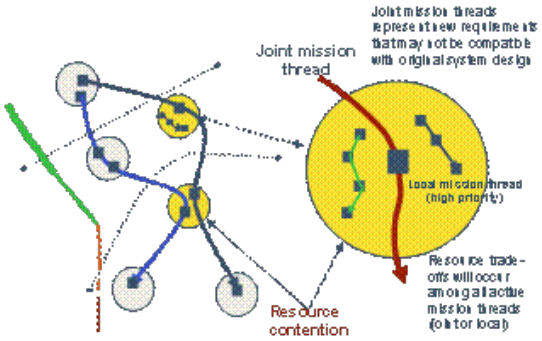

Figure 3. Joint threads introduce new requirements and conflicts.

Software-intensive, complex, human-machine systems require distributed decision making across both physical and organizational boundaries. The combination of expanding interoperability, multilevel requirements, and multiple control points creates a highly complex operational environment in which system behavior can be difficult to predict. An SoS mission thread requires reliability and dependability for individual components and solid end-to-end engineering so that the integration of those components ensures the survivability of the thread.

Seamless integration among cyberspace participating systems will require operational support and sustainment efforts to be able to recognize unacceptable conditions, resource contention among participants, and changes in resources with options to respond to the potentially bottlenecking situations affecting mission success. Both monitoring and response needs must be considered as systems and software components are designed and built since the network communication structure's controlling mechanisms provide a limited range of integration control in an SoS environment.

Challenges for SoS Operational Survivability

There is ample evidence that we are already reaching the limits of our engineering and testing practices, even in the today's less dynamic environments [Howard 06]. Unfortunately, systems engineered to work together commonly fail to produce the desired joint outcome because all circumstances in the operational environment could not be anticipated. In fact, even system upgrades can break existing interoperability among partnering systems.

Operational survivability, both today and in the future, can also be affected in following ways:

- The difficulty of predicting the behavior of an SoS is compounded by multiple

managerial and technical control points that do not necessarily share common risk mitigation strategies.

- The effects of a component change or failure are difficult to contain, and, given external interdependencies, they may be unknown. An insignificant local change or error can be magnified by unexpected or poorly understood interdependencies.
- The interdependencies and complexity can hide risks that may not be observed until deployment. Risks may be difficult to identify for new uses of existing systems or when systems are linked in unanticipated configurations.
- The requirements for building a system today are typically based on pre-defined organizational and user actions that are validated prior to implementation. The deployment requirements for an SoS participant or service are not as well defined. While the interfaces may be stable, the behaviors of systems of systems change as new services and participants are added and new usages of existing services are deployed.
- As complexity increases in both the system capabilities and the interconnections in an SoS, the ability of humans to manually monitor and respond in a sufficiently timely manner becomes less feasible. In addition, it is impractical to require manual changes at each application or to require redeployment for configuration and security changes.
- At any given time, a large system has some component involved with error recovery. For a complex operational environment, the effects of an error can be propagated to multiple systems. Error recovery is complicated by the decentralization of control and system evolution, and by inherently diverse, conflicting, and possibly unknowable requirements. That complexity can be offset by the capability of the infrastructure and its components to adapt to error states by using alternate pathways, execution sequences, and strategies.

Thus, the future systems-of-systems infrastructure will fundamentally alter the relationship between system components. Each component will know far less about the time, reason, and environmental conditions in which it is invoked. Components must assume that errors are occurring. To protect itself, and to continue to execute its missions, a component within the infrastructure must adopt a defensive posture against a wide range of potential complications (or stresses) that were most likely not predicted during its development. Survivability of the mission threads will depend on how components manage these stresses.

New Assurance Mechanisms

Survivability Analysis Framework

The Survivability Analysis Framework (SAF)[1] was developed to help organizations analyze and understand threats and gaps to survivability for operational mission threads within an SoS. A report will be published in later this year describing the details of SAF along with examples. In the meantime, this article provides a summary to prepare the readers for the future document.

SAF is designed to address the following:

- identify potential problems with existing or near-term interoperations among components within today's network environments
- highlight the impact on survivability as constrained interoperation moves to more dynamic connectivity
- suggest engineering guidelines to increase the chances for survivability of mission threads operating within the SoS infrastructure
- increase our assurance that the mission threads can survive in the presence of failures

Mission Thread Steps and Step Interactions

Each critical step in a mission thread is tasked to fulfill some portion of mission-thread functionality. This tasking represents a contract of interaction between the mission-thread step and prior and subsequent steps. Pre-conditions establish the information provided to the step. Pre-conditions may trigger the execution of a step (e.g., data or a

human command), or the process may be continually executed (e.g., a sensor). Each step will have outcomes (post-conditions) that may interact with subsequent steps. However, the contract with prior and subsequent steps is not necessarily static and may have to be negotiated at run time to reflect the current situation. Even the identity of prior and subsequent steps may vary across executions of the mission thread.[2]

Environmental, data, process, and interaction limitations can lead to potential breakdown of a step. Each limitation represents a source or type of stress on the step and, consequently, on the mission thread. However, such stress does not necessarily cause failure. Steps can be designed to manage a range of stresses and still respond appropriately or degrade gracefully. Additionally, the failure of any specific step may not necessarily doom a mission, because subsequent steps may continue to execute the thread.

Linkages among steps are driven by three primary components: people, resources (technology, systems, connectivity, etc.), and interactions (e.g. data exchange). The behavior of the linkages coupled with the activities to be addressed in each step can lead to stresses. Unmanaged stresses can potentially lead to complete failure. The mission also likely will fail if a step manages a stress in a manner incompatible with subsequent steps. For example, consider a step that receives some data as input. If the value received by the step is out of the expected range, then the step can respond in a variety of ways. For instance, it might substitute a default value in place of the out-of-range value. This substitution, however, may have dire consequences if the decision to manage the stress by substituting a default value is inconsistent with the subsequent step's expectation for a highly accurate value.

SAF captures for analysis the ways in which selected stresses are handled at critical mission thread steps. It also analyzes whether the stress-handling approaches adopted by a step are compatible with subsequent mission thread steps. SAF consists of two component groups: (1) three matrices that capture stresses on a step and potential mechanisms for managing these stresses; and (2) a process for applying the matrices to a joint mission thread.

Applying the Survivability Analysis Framework

To begin the process, select a business-process (mission-thread) scenario with sufficient complexity to cross a range of organizational and technical options useful for analysis. Establish appropriate completion criteria for the scenario to represent organizational success. Decompose the scenario into a specific sequence of end-to-end steps (unique activities) that must be performed to reach the success goals.

Across the range of mission steps, assemble a matrix for each of the following:

- people (roles) required to participate in each step
- resources required for the activities in each step (organizational policies and procedures, technology tools and capabilities, network connectivity, etc.)
- pre-conditions, activities, and post-conditions expected based on successful step execution as pictured in Figure 4

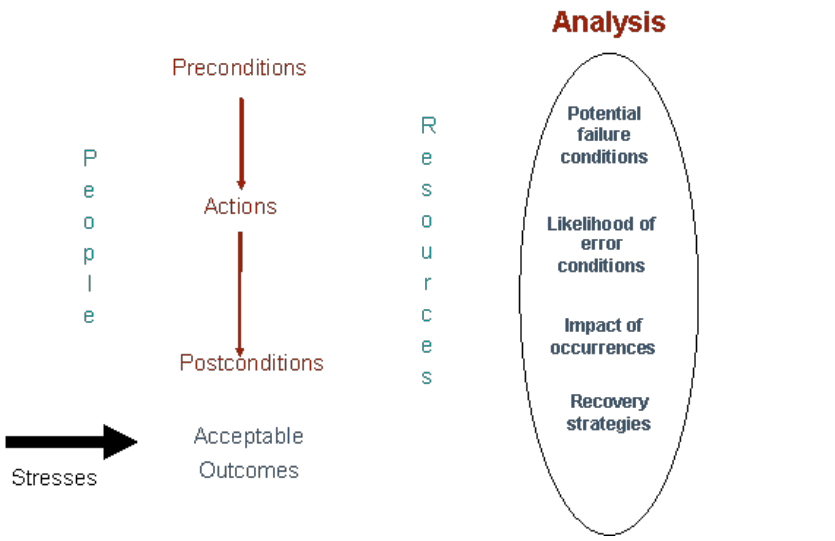

Figure 4. Survivability Analysis Framework

Using the three matrices, apply failure analysis techniques to identify potential points of failure that would critically impact successful completion of the mission thread [Alberts 05, Stamatis 03, Woody 07].

To begin to realize its potential, a broader range of operational mission threads must be analyzed with SAF. From a larger body of mission thread and development assessment examples, patterns of effectiveness can be identified and the framework refined.

References

[Alberts 03]
Alberts, C. & Dorofee, A. Managing Information Security Risks The OCTAVE Approach. Boston, MA: Addison-Wesley, 2003.

[Alberts 05]
Alberts, C. & Dorofee, A. Mission Assurance Analysis Protocol (MAAP): Assessing Risk in Complex Environments (CMU/SEI-2005-TN-032) Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2005.

[Howard 06]
Howard, M. & Lipner, S. The Security Development Lifecycle SDL: A Process for Developing Demonstrably More Secure Software. Microsoft, 2006.

[Leveson 04]
Leveson, N. “A Systems-Theoretic Approach to Safety in Software-Intensive Systems.” IEEE Transactions on Dependable and Secure Computing 1,1 (January-March 2004): 66-86.

[Maier 96]
Maier, M. “Architecting Principles for Systems of Systems” 567-574. Proceedings of the Sixth Annual International Symposium, International Council on Systems Engineering. Boston, MA, 1996. www.infoed.com/Open/PAPERS/systems.htm.

[Stamatis 03]
Stamatis, D.H. Failure Mode and Effect Analysis: FMEA from Theory to Execution, 2nd ed. Milwaukee, WI: ASQ Quality Press, 2003.

[Woody 07]
Woody, C. & Alberts, C. “Considering Operational Security Risk during System Development.” IEEE Security & Privacy 5, 1 (January/February 2007): 30-35.

1 SAF was piloted for the U.S. Department of Defense, Joint Battle Mission Command and Control (JBMC2) in analysis of a time-sensitive-targeting mission thread for the Office of the Undersecretary of Defense Acquisition & Logistics (OUSD/AT&L). A

second pilot analysis was completed for time-sensitive-targeting information assurance for the U.S. Department of Defense, Electronic Systems Center, Cryptologic Systems Group, and Network Systems Division (ESC/CPSG NIS).

2 Mission threads are expected to be dynamic in content because each specific mission is unique.

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 for Acquisition—The First New Constellation

NEWS AT SEI

Author

Mike Phillips

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

Process Improvement

CMMI

This article was originally published in News at SEI on: June 1, 2007

First, the Development

The History

The first effort to address acquisition best practices began shortly after the release of the Software Capability Maturity Model (SW-CMM). A team from government and the SEI created the Software Acquisition CMM (SA-CMM) in 1994. That model was used by a variety of Department of Defense (DoD) and other government agency teams for about a decade. During that time, it was updated twice. In 2004, the DoD directed the creation of a CMMI-compatible collection of best practices for acquisition as an acquisition module. This release of the CMMI-AM was a concise version for use within government program offices.

In 2005, work had begun on CMMI for Development (CMMI-DEV) V1.2, and the CMMI model architecture was updated. This update of the architecture allowed a more complete approach to process improvement in the acquisition processes of organizations. Initial leadership for developing acquisition improvement came from the chief information officer of General Motors, Ralph Szygenda. His vision was to improve how GM's regional centers acquired the critical software needed to manage GM's infrastructure around the world. The initial draft of this guidance, *Adapting CMMI for Acquisition Organizations: A Preliminary Report*, was published by the SEI in June 2006, just before the rollout of CMMI-DEV V1.2.

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 the past year, a combined team from government, industry, and the SEI has reviewed pilot feedback to develop the material and assure that it covers government acquisition as well as the outsourcing efforts typical in industry. The tested processes used for the CMMI V1.1 and V1.2 updates were reused, and an advisory group with rich acquisition experience reviewed proposed changes and the actual implementation. The result is the CMMI for Acquisition (CMMI-ACQ) V1.2 model. This model is now in draft form and has now entered the final quality-assurance phase of its development. Release of CMMI-ACQ is planned for November 1, 2007.

What Are the Major Differences Between CMMI-ACQ and CMMI-DEV?

As described in earlier columns, the constellation approach is designed to maximize the commonality of CMMI models while recognizing the need for both additional coverage and some differing process areas that better define best practices in the selected area of interest. Sixteen of the process areas appear to be suitable for all CMMI constellations. Material additions were allowed to these 16 process areas, but no deletions were allowed. The other six process areas have significantly different material from that in CMMI-DEV and were therefore renamed to recognize this distinction.

Additions to the CMMI Model Foundation (CMF)

The CMMI-ACQ team recognized two elements of project planning and project monitoring and control processes that are not covered in CMMI-DEV, but are particularly important to acquisition best practice.

The first of these is establishing an effective acquisition strategy. An acquisition strategy is often created early in a planning effort. A practice was therefore added early in Project Planning (PP) to cover this important acquisition activity.

The second of these elements is the transition of acquired products into use. Program offices frequently employ extensive efforts after product delivery to ensure customer satisfaction. After debating whether or not to create a new process area covering this topic, the team decided that adequate coverage could be achieved by adding specific practices to the Project Planning and Project Monitoring and Control (PMC) process areas.

These model updates result in an approach similar to the one in CMMI-DEV that addresses data management. The importance of cost and schedule tracking to acquisition projects also suggested an improved emphasis on this area in the model. The team decided to add a chart in the Measurement and Analysis (MA) process area to meet that need.

The final change to the 16 process areas deserves mention as well. One of the unique challenges in acquisition is the need to form effective teams across organizational boundaries. Often multiple products or systems must be acquired to satisfy a customer need. Teams often must coordinate the functions, manage the risks, and handle information flow as part of complex relationships with other organizations. The CMMI-ACQ team concluded that practices like those in the integrated product and process development (IPPD) addition in CMMI-DEV would be useful to address these challenges. Therefore, material intentionally similar to the IPPD material in CMMI-DEV was added to CMMI-ACQ and positioned in the Organizational Process Definition (OPD) and Integrated Project Management (IPM) process areas. The key difference in the new model is that these practices are considered expected, rather than an optional addition that can be selected, as it is offered today in CMMI-DEV.

How About the New Process Areas?

Below I provide a short discussion of each of the new process areas contained in CMMI-ACQ. However, it may be useful to recognize the influence of CMMI-DEV on these process areas, since most readers of this column are already familiar with the Development constellation.

It should come as no surprise that a major emphasis of CMMI-ACQ is to increase the attention to the area covered by the Supplier Agreement Management (SAM) process area in CMMI-DEV. While this process area was allowed to be not applicable for development organizations that do not have suppliers external to the project, these practices are the essential purpose of organizations addressed by the new acquisition

(< 5 minute) survey.

constellation. To give the greater granularity appropriate to this focus, two process areas take the place of what SAM covered in CMMI-DEV.

The first new process area, Solicitation and Supplier Agreement Development (SSAD), covers the activities required to get a supplier agreement in place. The second process area, Agreement Management (AM), covers the needed business aspects of the relationship between the acquirer and supplier over the life of the project once the agreement is in place. Thus SSAD and AM extend the coverage provided in SAM Specific Goals 1 and 2, respectively.

Three other process areas are closely related to CMMI-DEV Engineering process areas. Acquisition Requirements Development (ARD), Acquisition Verification (AVER), and Acquisition Validation (AVAL) share much with their CMMI-DEV counterparts.

The remaining process area, Acquisition Technical Management (ATM), covers the role of the acquisition organization when the developer is both developing the product or service and overseeing integration of products and services. Central to this role is providing appropriate technical reviews to check progress and resolve technical problems that are frequently part of the development of a complex software-intensive system. To resolve integration-related problems, practices that ensure good interface management are included. We think the addition of ATM will also help acquirers deal with today's increasing attention on systems of systems and network-centric needs in many acquisitions. This improvement in coverage is similar to the elevation of integrated teaming coverage mentioned a few paragraphs earlier.

Two differences from CMMI-DEV that I will mention here are that, based on pilot feedback, the team determined that ARD needed to be at Maturity Level 2 to provide the right emphasis on the acquirer's role in getting the requirements right as soon as possible. One other difference to note is that for this constellation, we put the six new process areas in an Acquisition process area category. We decided not to maintain the Engineering category with a single process area, Requirements Management, in it. Instead, we gave REQM a new home in the Project Management process area category.

What About Training?

The team has begun working on a course revision to reflect a multi-constellation approach for CMMI. At initial release of CMMI-ACQ, however, we will first offer a modular approach to the training similar to our transition from V1.1 to V1.2 in CMMI-DEV. Access to the Internet-provided module will require successful completion of the existing CMMI-DEV V1.2 Intro course, or its V1.1 predecessor. This course builds on the solid foundation in the workings of CMMI models given in the existing *Introduction to CMMI* course and describes the best practices that the acquirer should be seeking in his or her suppliers of choice.

Based on our experience providing CMMI-DEV to several acquisition organizations in the DoD, we will provide a limited number of offerings of a course that addresses both development and acquisition before release of the integrated course next year. Our course offerings will depend on community interest. We anticipate chartering an initial cadre of SEI Partners with experience in the field, and will add others consistent with community demand.

What About Appraisals?

The introduction of a significant variant to the established benchmarking model in SCAMPI Class A appraisals has caused us to take an approach different from previous CMMI releases. No SCAMPI Class A results using the CMMI-ACQ model will be accepted by the SEI for the first six months after release on November 1, 2007. Other classes of appraisals may be used to encourage process improvement progress during that period. We are also investigating ways to accommodate organizations that have significant parts of their organizations responsible for acquisition while other parts are responsible for development. We plan to provide an approach that maintains the needed confidence in appraisal results but recognizes the need to avoid the cost of two separate appraisals.

And for the Rollout …

In November, the SEI will announce CMMI-ACQ V1.2 and provide it in both PDF and

Microsoft Word formats on the SEI Web site. Over time, summary versions similar to the predecessor CMMI-AM V1.1 also will be provided on the same Web page. The existing CMMI V1.2 Tutorial will be updated to address the CMMI-ACQ constellation, and this updated version will be presented at various conferences over the coming year. The SEI will also provide a comparison document that shows the strong similarities between the CMMI-DEV and CMMI-ACQ models as well as the differences between these two related areas of interest.

Summary

We are excited about the coverage that this new constellation provides to focus on what some call “acquisition,” that others call “outsourcing,” and still others call “supply-chain management.” We also believe that there are improvements in coverage of topics, such as integrated teaming, that will, over time, suggest changes to the next version of CMMI-DEV. Together these two constellations, combined with the technical report *Understanding and Leveraging a Supplier's CMMI Efforts: A Guidebook for Acquirers*, offer significant progress in improving the systems we provide to our various end users.

About the Author

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.

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

Being Your Own Boss—Part III: Knowledge Work

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: June 1, 2007

This is the third in a series of columns on being your own boss. The first two columns discussed ideal jobs and autocratic management. In this column, I discuss knowledge work—what it is, and why it is so important. I also discuss why the management of knowledge work is fundamentally different from the way management is typically performed today and the consequences of today's common management styles. While I will ultimately talk about what you and I, the knowledge workers, can do about the way we are managed, I am not yet ready to discuss that in this column.

The key point is that how we are managed is up to us. Once we have acquired the requisite knowledge and skill, we really will have the ability to manage our own projects, even if our organizations and our immediate managers do not know how to manage knowledge work. However, to be successful at doing this, we will have to show our managers how to manage us. Understanding how to do that, of course, is the key, and that is what I am talking about in this series of columns.

The Typical Team Plan

In part II, I mentioned the way development plans are typically made today. The manager sits down by himself or herself and lists half a dozen or so checkpoints, gives them more or less arbitrary dates that in aggregate meet management's desired delivery date, and then reviews this "plan" with the team. After the team has commented on the plan, and the manager has made a few minor changes, this becomes the official team plan. The team is then committed to delivering the product on the date specified in this plan. The fact that none of the developers think that this plan has a reasonable delivery date or that they can meet the schedule is beside the point; this is the "official team plan."

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

(< 5 minute) survey.

There are three things wrong with such plans. First, the team does not own the plan; it is the team leader's plan. The members are not committed to it, and they do not believe they can meet it. Second, plans like this do not guide the work, and they do not provide a realistic estimate of how long the work will take. Third, development work is knowledge work, and plans that are produced in this way are not suitable for guiding knowledge workers. In addressing these points, I start at the back and work forward.

In this column, I discuss knowledge work, what it is, why it is important, and how to manage it. In subsequent columns, I will address the characteristics of plans that can truly guide knowledge work and help developers accurately commit to how long their projects will really take. After this important preparatory material, there will still remain the problem of getting people to actually work in this way. This is also a topic for future columns. In case there is any suspense, however, and for those of you who don't know me and my work, all of the columns in this series are leading up to a discussion of how the Team Software Process (TSP) can guide you as you strive to become your own boss.

Knowledge Work: Definition

Knowledge work is work with minds instead of hands. While your hands may be involved, the true product of knowledge work is concepts, ideas, and designs and not the devices, machines, or things that may ultimately be produced from these knowledge products. The knowledge worker is most productive when he or she is creating, refining, recording, or elaborating ideas, concepts, and representations. This is creative work, and it requires a very special kind of management. Peter Drucker originated the concept of knowledge work, and he discussed it extensively in his final books and papers [Drucker 1999]. He describes what knowledge work is and how to prepare for it, and his books and papers are well worth reading.

Managing Knowledge Work

My objective, however, is to describe how knowledge workers can take control of their own work. This is not a subject that Drucker has addressed. However, Drucker did succinctly state the management principle for knowledge work, which is that managers can't manage knowledge work; the knowledge workers must manage it themselves. I add to this the point that, for knowledge workers to manage themselves, they must know how to manage. While all of this may sound very logical, there are some key questions. First, why must knowledge workers manage themselves? Second, why don't they manage themselves today? And third, what is wrong with having managers manage knowledge workers?

First Question: Why Must Knowledge Workers Manage Themselves?

First, the reason that knowledge workers must manage themselves is that nobody else can. We all think in our own way and often we don't think much about how we think; we just do it. While this is a natural and comfortable way to work when you can, most of us work for organizations, and we are paid to produce results on committed schedules. However, to meet our schedule commitments, we must work to achievable schedules.

Since the essence of management is using the available resources to produce defined and committed results, the manager must know a great deal about the work, the available resources, and how the work is to be done. This means that whoever manages knowledge work must be able to project how long the work will take. To do this, however, that manager must also understand something about the products to be produced, how the knowledge workers intend to produce them, and how long similar work has taken them in the past. Finally, this requires that these managers have some historical data on how long it took these knowledge workers to do similar work in the past.

The reason that nobody but the knowledge workers themselves can manage knowledge workers is that their work is highly individual. If you ask designers how they design, for example, and assuming that they have thought about this question, they will tell you that they leap from one abstraction level to another, and frequently dive from the highest level all the way down to code and back [Curtis 1999].

When you couple this dynamism with the fact that you can't watch people and tell how they are thinking, it is obvious that nobody but the knowledge workers themselves can know what they are doing, how they are doing it, what they have accomplished, or how long it took. Since you must know all of these things to manage any kind of sophisticated work, this means that nobody but the knowledge workers themselves can manage knowledge work.

Second Question: Why Don't Knowledge Workers Manage Themselves?

This gets us to the second question: "Why don't knowledge workers manage themselves today?" The principal reason is that few knowledge workers know how to manage themselves. For example, the things that software developers must be able to do to manage themselves are the four things I just listed above. They must know what they are doing, understand how they are doing it, be able to measure what they have accomplished, and gather data on how long it took. While knowing how to do all of this is not terribly difficult, it is not obvious. In fact, explaining how to do this is the principal reason that I wrote several of my books. The most relevant one for today's software professionals is *PSP: A Self-Improvement Process for Software Engineers* [Humphrey 2006].

The basic requirements for self management are pretty straightforward, and they only require that the knowledge workers be able to convince their management that they can manage their own work. Then, of course, they must actually manage it properly and do it so consistently and well that management will continue to trust them to manage themselves in the future. This, however, is the key: trust. If your managers do not trust you to manage your own work, they must manage you. They may not want to, but their continued success as managers depends on the results that you produce.

Since your manager's performance depends on your performance, and since the performance of software groups has historically been so poor, managers do not trust software professionals to manage themselves. To overcome this problem, all we have to do is to convince management that we can manage ourselves and then perform that self management so well that management will continue to trust us. Since software groups have been unable to do this in the past, they have not been trusted to manage themselves. This situation will continue until we software professionals do something about it. For more information on how to do this, read the rest of this series of columns.

Third Question: What's Wrong with Having Managers Manage Knowledge Workers?

This now gets us to the third question: "What is wrong with having managers manage knowledge workers?" Here the answer has two parts. The first is that even though the managers don't know how to manage the knowledge workers, that doesn't mean they won't try. Unfortunately, however, when they do try, they do a bad job. That is why we get vague, inaccurate, and highly misleading plans and why software projects so frequently miss their schedule commitments. Furthermore, since the managers can't produce precise or detailed plans, and since there is no way that they could know how long the various knowledge-working tasks would take, there is no way to measure and track progress against these plans.

Also, as noted above, when the managers make the knowledge-workers' plans, the knowledge workers do not own these plans, and they have no real commitment to them. So, in summary, the reasons that managers should not manage, or even try to manage, knowledge workers are the following.

- First, the managers cannot possibly do a competent job of managing knowledge work.
- Second, when managers try to manage knowledge work, the plans they produce are so inaccurate and misleading that neither the workers nor their managers can measure or track project progress, and the work is unpredictable and usually late.
- Third, when knowledge workers work with such incompetent plans and plan management, they are unhappy and are most likely to do poor work.

That is why the managers should not manage knowledge work. However, the problem is

that, if the knowledge workers do not manage themselves, the managers must.

In the next column, I will start to address the question of what software developers and other knowledge workers must to do get control over their own work. Basically, this is the issue of building trust and credibility with management. Unfortunately, credibility and trust are hard to build but very easy to lose. All you have to do is to blow one schedule. This, of course, leads us to a circular problem: the managers blame the developers for missing their schedules and the developers blame the managers for giving them schedules that they can't meet. Breaking this circle is not a trivial problem, which is why so few knowledge workers are actually able to manage their own work.

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 Judi Brodman, David Carrington, and Bob Schaefer.

In Closing, an Invitation to Readers

In these columns, I discuss development issues and how they impact the work of engineers 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. Better yet, include a story or brief anecdote to illustrate 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

References

[Curtis 1999]
Curtis, Bill; Krasner, Herb; and Iscoe, Neil. "A Field Study of the Software Design Process for Large Systems," *Communications of the ACM*, November 1988, Vol. 31, No. 11.

[Drucker 1999]
Drucker, Peter F. *Management Challenges for the 21st Century*, New York: Harper Collins, 1999.

[Humphrey 2006]
Humphrey, Watts S. *PSP: A Self-Improvement Process for Software Engineers*, Reading, MA.: Addison Wesley, 2006.

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 11 books. His most recent books are *Winning with Software: An Executive Strategy* (2002), *PSP: A Self-Improvement Process for Software Engineers* (2005), *TSP, Leading a Development Team* (2006), and *TSP: Coaching Development Teams* (2006). 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. In a White House ceremony in 2005, President George W. Bush awarded him the National Medal of Technology.

Find Us Here

Share This Page

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

Mexican Advocates See TSP as a Way to Establish Quality Reputation

NEWS AT SEI

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

Process Improvement

TSP

This article was originally published in News at SEI on: June 1, 2007

For many years, the Team Software Process has developed dedicated practitioners who attest to its merits, and its growth into Mexico has expanded its base of enthusiasts. On their recent visit to the SEI, the people who helped spearhead that expansion expressed abundant confidence and conviction. As Mexico becomes a global force in information technology, Mexican advocates are certain that Team Software Process (TSP) adoption will establish Mexico's reputation for high-quality software and engineering.

Mexico is no newcomer to SEI process-improvement methods. The Capability Maturity Model Integration (CMMI) course was first delivered there in 2002. CMMI is a set of practices that can be used to guide processes across a project, a division, or an entire organization. It defines levels of maturity that reflect an organization's ability to implement its practices. While CMMI is highly effective and adopted worldwide, the complex requirements of CMMI implementation may cause some trepidation to small companies with limited resources. TSP can expedite this implementation and help organizations achieve higher CMMI levels in much less time than is usually required. This capability is especially suitable for Mexico, where 80% of software companies employ fewer than 50 developers.

Organizations widely use TSP to dramatically improve software development teams' productivity and reduce defects, costs, schedule deviations, and time to market. TSP works in conjunction with the Personal Software Process (PSP), through which individual engineers can measure and enhance their performances. The term *TSP* generally refers to the two processes as they are practiced in tandem. Both were created by Watts Humphrey as a way to bring CMMI principles to teams and individuals.

Related Links

News

Heartbleed: Analysis, Thoughts, and Actions

TSP Symposium 2014 Goes Beyond Methodology to Focus on Software Quality

See more related news »

Training

See more related courses »

Events

Team Software Process (TSP) Symposium 2014

Nov 3 - 6

Rafael Salazar was especially enthusiastic as he told his story of how TSP came to Mexico. Salazar serves as the director of the TSP initiative at Tecnologico de Monterrey, where he began classroom training in TSP. Salazar was looking for such a process several years ago. At that time he directed 500 software-development engineers for a large company that had adopted CMMI. He recognized that CMMI was valuable to organizations but believed that something additional was needed to bring its principles directly to developers. Salazar began reading Watts Humphrey's books on the Personal Software Process and continued to learn about PSP while heading the Computer Science Department at Tec de Monterrey. He became convinced that it was the missing element he had been seeking.

The advantages that TSP would bring to the Mexican software engineering community were obvious to Salazar. TSP would serve as a CMMI accelerator, as it had for many U.S. organizations. While the CMMI framework provides top-down guidance for *what* organizations should do to improve processes, TSP provides principles for *how* to implement most of the CMMI process areas. TSP application has helped organizations hasten their achievement of high maturity—for example, to reach Level 4 from Level 2 in 16 months instead of the average 50 months [Pracchia 04]. Used with or without CMMI, TSP provides major benefits for organizations wanting to dramatically upgrade performance to consistently meet deadlines and estimates, reduce costs and defects, and satisfy customer requirements.

Salazar visited the SEI in December 2005 with others from Tec de Monterrey and met with Jim Over, team lead for the SEI TSP group. In March 2006, Tec de Monterrey signed a memorandum of understanding with Carnegie Mellon to collaborate in a TSP initiative. By the end of 2006, Tec de Monterrey had

- obtained funding for the first phase of the project from the Mexican Ministry of Economy;
- conducted a TSP initiative launch with Watts Humphrey;
- conducted a faculty workshop;
- begun TSP training of software engineering undergraduates and faculty; and
- begun collaborative work with two companies in Mexico, Softtek and IBM.

This year has brought collaboration with a third company, Towa, and continued faculty and undergraduate training in TSP.

These are great strides toward the two main objectives of the TSP initiative: international recognition of the Mexican software industry as a high-quality industry, with high-quality human resources and high-quality projects; and the establishment of TSP capabilities through software-development companies, developers, instructors, and coaches.

Salazar is certain that the first goal will be realized through TSP. He readily recites the numbers regarding TSP benefits: It can accelerate CMMI introduction by 60% and brings 2 to 20 times improvement in product quality, 2 to 10 times reduction in testing time, a reduction in software defects to 60 per million lines of code, and job-estimation accuracy between +/- 10%.

Tec de Monterrey has made striking progress toward the second goal through its training of TSP professionals. As part of its software engineering curriculum, this program has achieved a rapid pace in training certified PSP developers, authorized PSP instructors, and TSP coaches. Currently being trained are 45 software engineering undergraduate students and 10 members of the software engineering faculty. Jim Over and Watts Humphrey have remained directly involved in nurturing this progress and ensuring proper standards for all certifications and authorizations.

Skills are being diffused across a fast-growing software-engineering community. Ivette Garcia, director of digital economy of the Mexican Ministry of Economy, describes her work with PROSOFT: National Public Policy to Enhance the Mexican IT Industry. In striving to develop a strong IT services industry, her organization has lent substantial support to the TSP initiative, as have the state governments of Jalisco and Nuevo Leon. She explains that since 2002, before the TSP initiative began, Mexico's IT productivity

had begun to expand considerably through a national effort involving PROSOFT, state governments, universities, and private industry. In the past two years alone, 11,000 new IT jobs have been added, and 122 new IT firms have emerged. Mexico is uniquely poised to become a global player in IT and software engineering. According to the McKinsey Global Institute's proprietary Location Cost Index Database [Garcia 07], among 10 countries positioned for providing IT services to the United States, Mexico ranks first in its business-environment attractiveness; ties with Brazil for first place in infrastructure quality; and is second only to China in its access to the U.S. market. But as excited as Garcia is about this amazing expansion, her main priority is quality.

In 2006 there were no PSP-certified individuals in Mexico. There are now 74—more than anywhere else in the world. The U.S. currently has 59; the other seven countries offering TSP have between one and three. Specialized skills, knowledge, and discipline set PSP practitioners apart from other engineers as they engage in many aspects of creating software: program development, requirements definition, document systems test, or maintenance and enhancement of large and small software systems.

This infusion of TSP expertise has important global implications. TSP-certified developers will provide a great human resource not only in Mexico, but also internationally. Their exceptional work habits and performance are bound to raise the standard of software engineering and software quality overall. TSP-trained developers, instructors, and coaches will reinforce this trend. Companies that engage their services will realize that defect-free software can be developed while decreasing cost and time to market.

References

[Pracchia 04]
Pracchia, Lisa. "The AV-8B Team Learns Synergy of EVM and TSP Accelerates Software Process Improvement." *CrossTalk: The Journal of Defense Software Engineering*. January 2004.

[Garcia 07]
Location Cost Index Database, McKinsey Global Institute.In *PROSOFT: Public Policy for the Mexican IT Industry*, presented by Ivette Garcia at the SEI, August 2007. Not publicly available.

Lessons Learned about Software Architecture

NEWS AT SEI

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

Software Architecture

This article was originally published in News at SEI on: July 1, 2007

"Architecture used to be a fairly ill-defined discipline. No one had the title of 'architect.' It was just a role that someone played—usually by the most experienced technical guys around and usually they were the oldest guys there—the ones who had experienced the most pain and been through the wars the longest. We all looked to them for guidance and that was architecture. Back then, it was truly an emerging discipline." That's how Jeromy Carriere, currently chief architect at VistaPrint, describes the art of software architecture in the 1990s.

Much has changed since then. Over the years, Carriere has worked in several industries as a software architect and learned many lessons about architecture and its importance to system success. During his keynote address[1] at this year's SEI Software Architecture User Network (SATURN) Workshop, he recounted these lessons and supported them with his own real-world experience:

- Big systems are hard to get right. Thinking about "architecture stuff" up front is necessary for success.
- Architecture is the bearer of quality, but reasoning about architecture is actually reasoning about the *potential* of a system.
- Performance and flexibility really do trade off against each other.
- Reprioritizing architectural qualities is extremely risky.
- Don't forget "saleability" and "marketecture." They can help you sell a system to upper management before it is built, even if they offend a "pure" architecture sensibility.
- Autonomy of organizations and systems is paramount, and this autonomy happens to be the foundational principle of service-oriented architecture.
- If you don't know where you're going, you're not going to get there—regardless of how good your map is.

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 »

- Technology doesn't matter. What does matter are people, the process, and the consistency of practice.
- Architecture validation is critical but hard to institutionalize—even in a process-oriented organization.
- The deepest problems in IT are still communication and understanding.
- Don't let "pragmatism" become a disguise for shortsightedness.

Carriere sees continuing potential in architecture reconstruction, seeing it as analogous to an archeological dig. "Archeology is when guys go out and dig stuff up, find artifacts, and then try to posit hypotheses about what human process created them. How did each artifact get there? And in fact not just the human process but also what environmental process contributed to producing that artifact the way it is, where it sits, and how it works. Software architecture reconstruction is exactly the same. We go into the code and we dig around and try to figure out how what design decisions produced that shadow of a process."

Through his work with architecture reconstruction and working on multiple and varying types of systems, Carriere has found that reasoning about architecture is really reasoning about the *potential* of a system based on that architecture. "I can take a great architecture, analyze it, prove that it has all these great qualities, but then somebody goes off and builds it, and it's in that building process that we often lose our grip on what it was we we're trying to achieve. That's why architecture reconstruction and architectural conformance are so important. They can give us the connectivity—or traceability—from design decision to execution that we need to make architecture truly valuable." Carriere's experience has also taught him that, because software architects eventually move on and find other roles, companies must find some thread of consistency through the systems they build. If they don't, organizational scalability is fundamentally threatened.

According to Carriere, the four areas of the SEI's current architecture research—systems of systems, architecture-based evolution, architecture competence, and economics-based architecture—respond to all of these lessons and to the issues faced by software architects in the field today.

Benefits of that research are working for people all over the world—at organizations like Ericsson, Samsung, and Sandia National Laboratories. At SATURN 2007, employees from those companies explained how SEI technologies and methods are working for them: at Ericsson, the SEI Quality Attribute Workshop (QAW) and work in tactics, patterns, and software architecture documentation; at Samsung, SEI Attribute-Driven Design; and at Sandia, the QAW. While some organizations—such as the U.S. Army—reported using the SEI Architecture Tradeoff Analysis Method® (ATAM®) as-is to evaluate their architectures, others like Raytheon, Wells Fargo, and Microsoft have customized the QAW and the ATAM to create unique evaluation methods. During a panel discussion on using the ATAM, Raytheon personnel presented ATO Lite—a method consisting of a subset of ATAM activities—that helps architects develop robust architectures in a timely and cost-effective way. And a Wells Fargo architect described how the company has leveraged both the QAW and the ATAM to create a hybrid evaluation method for exploring quality attribute concerns when a full ATAM is not possible. Carriere described an ATAM-related method that he developed while at Microsoft—the Lightweight Architecture Alternative Assessment Method (LAAAM). The LAAAM incorporates the scenario-driven perspective of ATAM into a lightweight approach to assessing high-level architectural decisions.

Workshop attendance doubled from 2006 to 2007—a trend that exemplifies the growing importance of architecture around the world. Many participants came from outside the United States—from Ireland, South Korea, Mexico, Australia, Japan, and Britain.

SATURN 2008 is scheduled for April 28 through May 1 in Pittsburgh, Pennsylvania. Planned topics include case studies and lessons learned in applying SEI software architecture methods, improving the state of architectural practices across an organization, evolving an architecture along with business and mission goals, and

future directions in software architecture technology.

This year's organizers are looking for up-to-date information on how companies are using both SEI and non-SEI methods and technologies to improve their software architecture practices and competence. If you would like to tell your story at this year's workshop, submit your presentation or tutorial proposal at http://www.sei.cmu.edu/architecture/saturn/. The deadline for submissions is December 7.

1 To see the slides from his keynote presentation, go to the library.

Mitigating the Risk of Using Service-Oriented Architectures

NEWS AT SEI

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

Software Architecture

This article was originally published in News at SEI on: July 1, 2007

Most of the practitioner-oriented materials about service-oriented architectures (SOAs) build on the SOA hype and focus on the benefits of SOA in terms of achieving business goals. Those benefits are real and substantial, but what about the many design decisions architects must make that affect the system's performance, interoperability, security, and modifiability? What can architects do to mitigate the risk of failing to meet those quality attribute requirements? According to the Software Engineering Institute (SEI), the answer is an early architecture evaluation.

In their technical report titled *Evaluating a Service-Oriented Architecture*, SEI researchers Paulo Merson and Phil Bianco, along with Rick Kotermanski of Summa Technologies, describe SOAs and discuss SOA design considerations and tradeoffs that can help the architecture evaluator identify and mitigate risks in a timely and effective manner.

There are many definitions of SOA, but none is universally accepted. However, according to researchers at the SEI, one central notion is common to them all—that of *service*. An ideal SOA service

- is self-contained
- is a distributed component
- has a published interface
- stresses interoperability
- is discoverable
- is dynamically bound

From the point of view of a software architect, SOA is as an architectural style where systems consist of service users and service providers. Because SOA involves multiple

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 »

systems, business entities, and technologies, its overall complexity and the political forces involved need to be factored into architecture tradeoff considerations more than in single-application designs where technical concerns predominate. One method that can be used to conduct SOA evaluations without any adaptation is the SEI Architecture Tradeoff Analysis Method(ATAM)—a method for evaluating software architectures relative to a system's quality attribute requirements. In addition to revealing how well an architecture satisfies particular quality goals, the method provides insight into how they interact with each other—that is, how they *trade off* against each other.

To guide evaluators and help them identify and mitigate risks to their systems, researchers at the SEI have compiled information about the architectural approaches and design considerations and tradeoffs that are particularly relevant to SOA. Because architectural design decisions determine a system's ability to meet functional and quality attribute goals, asking the following questions when designing an SOA can be especially helpful:

- **What communication mechanisms are used?** Service users and service providers can communicate via
  - Simple Object Access Protocol (SOAP) Web services
  - messaging systems such as Microsoft MSMQ and IBM WebSphere MQ
  - a simplistic but innovative approach called Representational State Transfer (REST)
  - a mix of these alternatives and proprietary communication protocols, varying from Systems Network Architecture/Logical Unit (SNA/LU) 6.2 to Internet Inter-ORB Protocol (IIOP)

  The right choices will help to achieve the desired interoperability, modifiability and performance.

- **Is an ESB part of the solution?** The Enterprise Service Bus (ESB) is a component that intermediates the communication between service users and service providers. The ESB can route, split, and merge messages; transform data's content and format; verify authorization; and adapt messages to different platforms/technologies. The gains in terms of modifiability, interoperability, and extensibility have to be weighed against the performance overhead and the initial cost and complexity of implementation. For smaller systems with a more homogeneous platform and a slower pace of changes in business and technology, direct point-to-point integration among service users and providers may work better than using an ESB.
- **What is known about the target platform?** In SOA solutions, many quality concerns are primarily handled or strongly affected by the runtime infrastructure. For this reason, architects should be familiar with the target platforms, including the runtime environment for service users and service providers, the network infrastructure, and the platforms used by external services.
- **Should a service be synchronous or asynchronous?** In an SOA, services may be provided through either synchronous or asynchronous interfaces. Each option has pros and cons to consider, and the selection of a service interaction approach depends on a combination of business and application logic requirements, existing component capabilities, and other architectural factors.
- **Should a service have a coarse- or fine-grained interface?** A coarse-grained service interface typically consists of operations that require less communication and are designed to do more work with fewer service calls than fine-grained services. Service interface granularity has architectural and business implications and is a critical factor for achieving performance, reusability, and modifiability requirements when implementing an SOA. Designing a service that is "right" grained depends primarily on how the service will be used, but the architect should also consider which quality attributes are most important to system stakeholders.
- **What strategies are being used for exception handling and fault**

recovery? Achieving reliability, availability, and serviceability requirements is difficult in SOA systems, because they sometimes involve heterogeneous platforms and protocols, as well as external services. A robust SOA-based architecture must deal with application and system failures at a variety of levels, such as the system infrastructure, networking, and data services. Establishing proper debugging, logging, and tracing components and related standards helps to detect failures and identify potential sources. Error-handling strategies can also manage the behavior of the system under failure modes.

- **Does the SOA involve https or message-level security?** Https is a simple and popular alternative to protect the communication pipes at the transport level in SOA solutions. However, it doesn't protect the message when it is processed by intermediary components. Message-level security provides an end-to-end solution that protects the message itself, but there are interoperability concerns.
- **How is service authentication and authorization managed?** Authentication and authorization in SOA solutions involve several design decisions regarding
  - access-control mechanisms (e.g., digital certificates, declarative or programmatic authorization)
  - technology (e.g., Lightweight Directory Access Protocol [LDAP]) and scope (e.g., enterprise wide, local) for security domains
  - access-control data representation format (e.g., Security Assertion Markup Language [SAML])
  - conformance to regulations (e.g., Health Insurance Portability and Accountability Act [HIPAA], Sarbanes-Oxley)

    These design decisions affect quality attributes such as interoperability, performance, modifiability, usability, and, evidently, security.

- **Is XML optimization being used?** XML is the most common format for data representation in SOA solutions. It is flexible, extensible, widely adopted, and the underpinning for interoperability in most SOA technologies. However, parsing, validating, and transforming XML data are memory- and CPU-intensive operations that can deeply impact the scalability and performance of the system.
- **Is a service registry being used?** In larger and rapidly changing SOA environments, it is difficult to manage the availability, capabilities, policies for use, and location of shared services. This difficulty results in the risk of quality failures. An SOA service registry provides the registration of services, management of metadata, and automation for the creation of and access to services.
- **How are legacy systems integrated?** There is typically more than one reasonable way to integrate a legacy system into an SOA environment. The architect must weigh the associated cost/benefit tradeoffs when selecting the integration strategy.
- **Is Business Process Execution Language (BPEL) used for service orchestration?** BPEL is an XML-based language for describing business process workflows that involve the interaction with multiple services. A BPEL engine executes code written in BPEL and coordinates the invocation of the services in the workflow. There are modifiability, interoperability, performance, and reliability implications both for using BPEL and for not using it. The architect must consider those implications when deciding whether to use the language.
- **What approach is used for service versioning?** Services and even individual operations within a service can be versioned independently of other system components. When the service is used by an unknown number of external service users, a common requirement is for old and new versions to coexist. That requirement makes configuration management and deployment more complex.

Because decisions about SOA tend to be pervasive and have a significant and broad impact on business, performing an early architecture evaluation is particularly valuable

and highly recommended. Balancing SOA aspects against other software architecture concerns is particularly challenging in an SOA software architecture evaluation.

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

Being Your Own Boss—Part IV: Being a Victim

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: August 1, 2009

This is the fourth in this series of columns on being your own boss. In the first two columns, I discussed ideal jobs and autocratic management. Then, in part III, I talked about knowledge work and why only the knowledge workers can competently manage their own work. The key question remaining at the end of part III was: How can we take charge of our own work? That is the topic I cover in this and the final columns of this series.

Software Developers Cry a Lot

When I ask software developers about the issues they face and the problems that typically cause their projects to fail, I hear lots of complaints. There are many reasons why our projects fail, and we typically blame somebody else for the failures. Management gave us an impossible schedule, the customer changed the requirements, the organization has an impossible bureaucracy, or there are too many meetings and distractions. This is victim talk. Have you ever heard a first class surgeon, a top flight scientist, or a winning ball player talk like this?

Winners win; they don't complain. It is the perpetual losers that complain about how unfair life is and how somebody else is always to blame for their failures. While it is true that software development is a challenging business and that we almost always face tight schedules and changing requirements, these problems can be managed. However, they can be managed only if you know how to manage them.

Taking Charge

You might wonder why more software professionals don't manage themselves, and why essentially all software developers act like victims. These smart and capable

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 seem willing to spend their lives behaving like losers. The principal reason is that nobody ever showed them how to break out of their victim trap. The second and almost as important reason is that, even though self-management methods are not that difficult, they are not obvious.

The alternative to being a victim is to take charge. There are two parts to doing this. The first is the hard part: actually taking control of your own work. Then, once you know how to manage yourself, the second part is much easier: convincing management to let you manage yourself. The rest of this column discusses the first part, and in the next column, I describe how to convince management to let you control your own work.

Managing Yourself

There are five principal steps to managing yourself, but they are not necessarily followed in linear order. Since all of these steps are interdependent, you must treat the entire process as cyclic. The principal reason to follow a cyclic strategy is that you will almost certainly make mistakes the first time, and with a cyclic strategy, you can quickly identify and correct these errors. However, you will be surprised at how quickly you will learn and how, assuming that you have a little guidance, you will do a good job of managing yourself the very first time you try. If you stick with the self-management process, your performance will quickly improve. In brief, the five basic steps to self management are described in the following paragraphs.

Step One: Establishing Process Goals

It is important to be clear about your goals and to make sure that these goals are ones that you are willing to strive to meet. With a little reflection, the importance of goals is obvious: if you haven't decided where you want to go, you are unlikely to get there. Examples of process goals are "meeting commitments" or "building high-quality products." If you don't feel that goals like these are truly important, you will not likely strive to meet them. And then, of course, you will almost certainly not do so. The other part of establishing goals is to make sure that they are consistent with the problems you are trying to solve. That means when trying to demonstrate to management that you can manage yourself, your goals must relate to the self-management problem.

Step Two: Define Your Process Principles

It is also important to be clear about the principles you will follow when using the self-management process. For example, one key principle is that it is always faster and cheaper to do the job right the first time than it is to build a poor-quality product and spend a lot of time fixing it. Another way to state this principle is that high-quality software products are always cheaper to develop and test than poor-quality products. Since few software developers truly believe this principle, it may not be one that you are willing to follow. Once you actually do follow it, however, you will be surprised to find that it is indeed true.

Another important principle is that, to meet commitments, you must plan and track your own work. An extension of this principle is that, to make accurate plans, you must know how long similar work has taken you in the past. This principle also relates to the first principle because poor-quality work is inherently unpredictable. While few software professionals will find this principle obvious, most experienced developers know that unplanned software work is rarely if ever delivered on time. The reason is that the key to making commitments is to only make commitments you can likely meet. And, of course, the key to making commitments that you can likely meet is to accurately estimate how long the committed work will take.

While there are many possible principles, it is important to be clear about the principles you intend to follow when you define your work processes, practices, and plans. Then, of course, you must consistently follow these principles while doing the work.

Step Three: Define Your Process and Plan

To consistently do competent work, you must define and document the process and plan you will use to do the work. While doing this, start the process with a planning step and conclude with a postmortem. Also define the measures you will use to track

your work as you do it. This is essential so that you can analyze your performance during the postmortem and then have the data you need to make progressively better estimates in the future. Since you will need this information for the very next cycle of the job, you must start gathering it right away.

The process you define is not some vague document to be put on the shelf; it is a working document to use in doing the work. That means that it must be an operational process. That is, it must be sufficiently detailed and precise to guide your work. Only then can you measure and estimate the work, track your process, and know if you are on schedule every day. Furthermore, the only way to consistently meet commitments is to work every day to recover from whatever problems you had the previous day. To do this, you must produce a plan with estimates and schedules for every step. For more information on defining operational processes, see my latest PSP book [Humphrey 2006].

The second part of step three is to actually produce a plan for the work. Once you have a little historical data and an operational process, the planning task is straightforward [Humphrey 2006]. If you don't have historical data, you will have to make some guesses. Usually, however, once you have used a defined and measured process to write even a few small programs, you can accurately tell how long such work has taken in the past. Then you can use these measures to guide the estimates for the next job. However, if you don't have any data and, since unmeasured recollections are generally inaccurate, you must be cautious about over-committing. Again, my PSP book can help you in doing this.

Step Four: Negotiate Your Commitments

In starting any job, use your plan to establish and negotiate your commitments. This, of course, is the bottom line. The key point to remember is that unless you make your commitments properly, you will rarely be able to meet them. That is why so many software projects get into trouble: few software professionals know how to make realistic commitments or to negotiate them with management. While negotiating commitments is not as hard as it might sound, it is not a trivial step. I will discuss it in more detail in the next column.

Step Five: Follow the Process and Plan

In doing the work, you must follow the process, gather the data, and use the data to plan and track this and all of your subsequent work. While this may sound simple, it is not easy to do, particularly the first time. That, however, is the reason that I developed the PSP process and course [Humphrey 2006]. While PSP takes a little effort to learn, it is really not difficult, particularly for competent programmers. A few people have completed PSP training on their own, but most find it is best to take a PSP course. The SEI offers such courses, but another and often more practical way to learn it is to convince your organization to have a qualified PSP instructor train you and your coworkers in PSP methods.

Conclusion

The purpose of this discussion is to explain why software professionals don't manage themselves and to convince them to do so. The simple reason that few developers do this today is that, while the methods are relatively straightforward, they are not trivial and they are not widely taught in a typical software engineering education. That is a shame, because understanding and following these principles will transform your life. Instead of being a victim, you can actually manage your own work. And that is the key to turning any job into an ideal job. The next column concludes this series on being your own boss with a discussion of how to negotiate with management.

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 Eugene Miluk and Said Nurhan.

In Closing, an Invitation to Readers

In these columns, I discuss development issues and how they impact the work of engineers 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. Better yet, include a war story or brief anecdote to illustrate 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

Reference

[Humphrey 2006]
Watts S. Humphrey, *PSP: A Self-Improvement Process for Software Engineers,* Reading, MA.: Addison Wesley, 2006.

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 11 books. His most recent books are *Winning with Software: An Executive Strategy* (2002), *PSP: A Self-Improvement Process for Software Engineers* (2005), *TSP, Leading a Development Team* (2006), and *TSP: Coaching Development Teams* (2006). 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. In a White House ceremony in 2005, President George W. Bush awarded him the National Medal of Technology.

Bright Star Dimmed: Roger Bate, 1923–2009

NEWS AT SEI

This article was originally published in News at SEI on: August 1, 2009

Roger Bate

"Texas Instruments had recruited Roger Bate to establish a software engineering group," recalled Larry Druffel, who was SEI director and CEO from 1986 to 1996. "Roger would recruit and train people in good software engineering practices for TI. And then they often were transferred to other TI programs. I recall visiting him to discuss TI software engineering practices, and he chuckled and said, 'A lot of my engineers get recruited away by other TI programs. That's all part of my strategy. I want them to do that, but I fight them every time, so they won't catch on.' That was the way he seeded good software engineering throughout TI."

Bate's later work with the Software Engineering Institute was instrumental in seeding best practices for software engineering worldwide. Bate, who was a former chair and member of the SEI Board of Visitors, and a senior member of the technical staff and, later, a visiting scientist at the SEI, died March 18, 2009, in McKinney, Texas. He was 86.

Born in 1923 in Denver, Bate began college at Cal Tech as a chemistry major studying under Linus Pauling in 1941. In 1943, in the middle of World War II, he enlisted in the U.S. Army Air Corps and transferred to the U.S. Military Academy at West Point. He graduated in 1947 and spent the next three years studying as a Rhodes Scholar at the University of Oxford, UK, where he earned degrees in nuclear physics. From 1951 to 1952, he served with the 10th Engineer Combat Battalion in Korea.

It *Is* Rocket Science

There is a common expression now, meant to express how simple something is: "It's not rocket science." But Bate's early work *was* rocket science, and he proved to be one of the brightest stars in a galaxy of brilliant lights.

As a young Army captain, he was sent in 1959 to be an instructor at the U.S. Air Force Academy in Colorado Springs, Colo. His assignment, he explained later, was to help set up the Department of Astronautics there. He transferred to the Air Force and became the first permanent professor of astronautics at the Academy in 1962 and the department's first head in 1963. He took leave from the department from 1963 to 1964

to earn a doctorate in aeronautical and astronautical engineering at Stanford.

Larry Jones, of the SEI's Research, Technology and System Solutions Program, was formerly chairman of the Computer Science Department at the Air Force Academy. "I didn't know Roger at the Academy because he had already left before I started there," said Jones. "But he was legendary. And he was legendarily brilliant."

Druffel concurred: "Besides Watts Humphrey of the SEI, I've worked with three other National Medal of Technology winners, and, intellectually, Roger was in their league."

"He was smarter than all the rest of us," agreed Jack Ferguson, a former faculty member in the Department of Astronautics at the Academy and the former head of the SEI's Appraisal Program, now a visiting scientist at the SEI. "I taught out of Roger's book when I was at the Academy," he said.

"Roger's book" is *Fundamentals of Astrodynamics*, which Bate co-wrote with Donald Mueller and Jerry White. The textbook was first published in 1971 and, astonishingly for a technology text, is still in print and still used in college courses and by professionals today. Internet reviewers of the book continue to hold it in high esteem: It is known as *BMW* (after the authors) by its devotees, and one even noted that he likes to keep two copies—one in his office and one in his briefcase.

What explains the staying power of a textbook nearly 40 years after publication? "I think the authors nailed it," explained Ferguson. "It has the right blend of mathematical rigor yet lack of complexity, so students can understand it."

Perhaps it was the need for mathematical rigor that led Bate into computer science. In the early days of the Academy, explains Ferguson, the Department of Astronautics had the greatest need for mathematical computing power. So the department members were the biggest users of the Academy's computers and needed to develop the programming standards. As head of the department, Bate established the computer science program and major at the Academy, which, at first, were part of the Department of Astronautics.

After holding other Academy positions, including vice-dean of the faculty, Bate retired from the Air Force in 1973 with the rank of colonel—in retirement he was later promoted to brigadier general—and left the Academy for Texas Instruments.

In a nearly 20-year career at TI, Bate formed TI's Advanced Software Technology Department and served in the Computer Science Research Department. Bate's interest in process improvement started in the late 1970s when he was leading the Advanced Software Technology Department and was tasked to solve "the software problem." This phrase was used in the defense industry to describe the problems that project managers would cite when trying to explain why their projects were over budget or late. Working with Edith Martin, who was the Deputy Under Secretary of Defense for Research and Advanced Technology, and Druffel, who was then the director of computer systems and software in Martin's office, Bate helped to organize a workshop that brought together 300 experts to study the issue. One of the results of that workshop was to validate the idea for the Software Engineering Institute. Druffel would later become SEI director.

Bate retired from TI in 1991 as chief computer scientist and TI fellow.

Joining the SEI

"In those days," explained Druffel, who is also a former faculty member of the Air Force Academy's Astronautics and Computer Science Department, "retirement at age 65 was mandatory at TI. Roger had joined the SEI Board of Visitors and was serving as chair when he called me one day, and he said, 'I have to retire but I'm not ready to quit working yet.' So I asked him to come to the SEI. He stepped down as chair and joined the SEI technical staff as an adviser to the director's office."

Druffel provided budget support for 25 percent of Bate's time; other SEI technical programs—if they wanted his expertise—would have to support the other 75 percent. Within a short time, Bate was 100 percent funded. "I worried that he was failing retirement and apologized one day to his wife Madeline who said, 'It is saving his life

to stay professionally active.'"

His work at the SEI included the initiation of the Systems Engineering Capability Maturity Model (SE-CMM). "The SEI could not have pulled that together without Roger," said Druffel, "because he came with a background in astronautics and aeronautics and TI systems. He was the only one at the SEI who had the credibility in systems engineering to lead that effort and did it effectively."

As Bate was working on the SE-CMM, other Capability Maturity Models were beginning to grow, said Druffel, who asked Bate to chair a group charged with evaluating the idea of integrating the CMMs in 1994. This work led to the Capability Maturity Model Integration (CMMI).

"There was a raging debate at the time," Druffel recalled, but he says Bate was the "right person to sort that out."

"He was an intellectual giant," Druffel said, "but he was the kind of person whose contributions have often been behind the scenes pushing other folks. He just had a way of identifying good people and helping them mature their ideas and follow through on them. So a lot of his contributions are of an intellectual nature that don't get the same kind of recognition, but they're no less important."

Sandra Shrum, co-author of *CMMI: Guidelines for Process Integration and Product Improvement, 2nd Edition*, said, "Roger's official title was CMMI chief architect, but he was also sometimes the chief diplomat. A lot of highly intelligent and very passionate people were working on CMMI, and that sometimes led to very passionate disagreements. Whenever this happened in a meeting or working session, as soon as the voices died down, everybody would look at Roger and just expect him to say the right thing—to help us move forward."

Jones, the computer science head at the Academy, noted the same quality: "I was on loan for the CMMI project. I got to observe his style first hand and was struck that a guy who was so legendarily smart was not only smart, but also accessible and down to earth. His leadership style was very casual. He would definitely form opinions, but he would also listen to feedback."

"I spent two days working with Roger at his house in Plano," Jones continued, "and he was just so gracious. He worked in a hands-on collaborative fashion on the details. We had to work out some thorny issues—some disagreements on fairly fundamental things. The models we were trying to merge were sometimes purposefully different. So to have brought people with differing viewpoints under the tent as part of the solution was a brilliant strategic management stroke. Then we just let the good work and the wisdom of the people involved come around on the working level as opposed to the political level. And Roger was the chief voice of reason in his position."

The culmination of Bate's work at the SEI came around the release of CMMI for Development (CMMI-Dev) V. 1.2. The CMMI Steering Group decided that a new architecture would help CMMI expand its coverage and extend into other domains. The revised version employs a new architecture and a new term—*constellations*—to describe how components are grouped. That description should not come as a surprise to anyone who knew Bate.

Bate gave some thought to future needs of CMMI users and how the architecture could meet those needs. "I thought of CMMI's collections of best practices as the stars of process improvement," he told a writer for the SEI Annual Report in 2006, "and I pushed the metaphor a little further to call a collection of model, training, and appraisal components for an area of interest a constellation." The metaphor came naturally to the former astronautics professor.

Summing up the contributions Bate made to CMMI and to the SEI, William Peterson, director of the Software Engineering Process Management Program, said, "Roger led the SEI's process work into the realms of systems engineering and integrated product and process development. With Roger's leadership, and with his collaborators' significant experience and expertise, we were able to greatly expand and enhance the improvement opportunities available to the U.S. defense community and to the

worldwide development community through CMMI. Throughout his life, Roger made a lasting, positive impact wherever he was involved."

SEI Director and CEO Paul Nielsen called Bate "a significant member of the SEI team who was instrumental in the creation of CMMI and its new framework. He was a friend and mentor to everyone who worked with him."

Nielsen, an Air Force Academy graduate and a retired Air Force major general, also noted, "Some of us who attended or taught at the Academy remember Roger as the head of the Astronautics and Computer Science Department. About a year ago, as the Astronautics Department celebrated its 50th anniversary, then Secretary of the Air Force Mike Wynne, who was previously a member of the astronautics faculty, promoted Roger to brigadier general for his contributions—a very rare, but richly deserved honor."

Bate was an SEI fellow, a fellow of the Association for Computing Machinery, and a fellow of the Society for Design and Process Science.

In addition to his wife, he is survived by two brothers, eight children, 11 grandchildren, and 3 great-grandchildren.

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

Clarifying the CMMI for Development Model for High Maturity

NEWS AT SEI

Author

Mike Phillips

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

Process Improvement

CMMI

This article was originally published in News at SEI on: July 1, 2008

The CMMI Steering Group and the SEI have agreed that the material contained in Version 1.2 of the CMMI for Development (CMMI-DEV) model should be clarified to better explain high maturity. These clarifications will be designed to address the needs that CMMI users, appraisers, and instructors have expressed related to CMMI high maturity practices. This column describes the reasons and current planning for how high maturity will be clarified.

What will be clarified?

Part of the purpose of the Version 1.2 release was to strengthen the integrity of CMMI appraisal results. An important aspect of this purpose included high maturity and capability (i.e., levels 4 and 5). Actions taken as part of Version 1.2 included the following:

- A certification process for high maturity lead appraisers was established.
- An appraisal lifetime policy was established, which limited the viability of appraisal results to three years.
- Rigorous reviews of appraisal results were defined and required to enable public announcements of appraisal rating results.
- Resources were committed to allow audits of all appraisals targeting level 4 and 5 appraisal ratings for several months. These audits have helped us assure that the fundamentals for CMMI high maturity are in evidence in the appraised

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

organizations claiming to achieve these levels. More information about the policy regarding high maturity appraisal audits can be found at www.sei.cmu.edu/appraisal-program/appraiser-communications/index.html.

The SEI hired two experienced lead appraisers to augment its quality assurance efforts early in 2008 and began conducting dedicated high maturity audits in February. As of October 2008, the SEI has conducted 29 audits of high maturity appraisals. These audits have been conducted both onsite and remotely. The time differences involved in some appraisals conducted half-way round the world from the audit team adds to the challenges faced by both the appraisal team and the audit team.

Remote audits involve an audit team that requests and receives appraisal information on high maturity elements from the appraisal team as the appraisal is being conducted. These high maturity elements are strong indicators that the organizations (and the high maturity lead appraisers) fully understand the fundamentals of high maturity and are properly using the tools needed to achieve high maturity and high capability (i.e., statistical management and process improvement models).

Audits are difficult to conduct, but are worth the trouble because of the valuable information collected. The valuable information received from those audited has confirmed that these audits serve a useful purpose of maintaining the integrity of appraisal results. We will soon shift our focus from auditing all high maturity appraisals to a sampling approach. This approach will assure that we do not focus exclusively on high maturity. There is a much greater number of appraisals aimed at achieving maturity levels 2 or 3 and others that focus on achieving capability profiles, without seeking maturity level equivalence.

Usually, audits confirm the findings of the appraisal team. Such audit results not only confirm the appraisal results for the organization, but also confirm the process and judgment used by the lead appraiser and his or her team.

Occasionally, audits uncover appraisal deficiencies. Examples of deficiencies include errors in judgment by high maturity lead appraisers and the evidence provided by the organization being appraised. In such instances, audit teams work closely with lead appraiser to ensure that the appraisal is brought to an appropriate conclusion. The outcomes of audits in these situations have resulted in the decertification of some high maturity lead appraisers, and in some cases rejection of submitted appraisal results.

So what is the plan?

The CMMI Steering Group and the SEI have agreed that they want to provide more help for users targeting level 4 and 5 appraisal results. Although there are white papers now available on the SEI website that provide clarification of high maturity practices, all agree that more is needed.

The first initiative is to create a set of criteria that describes the particular elements of the appraisal that are being audited. We think that communication of these criteria can help alleviate fear of the unknown that all audits can cause. The initial criteria will be identified with leadership from industry and approved by the Steering Group to assure that the breadth of model interest and experience is represented. As with other CMMI elements, the SEI will provide a change request mechanism so that further improvements can be made.

Work is also being done to craft improvements to the text of the CMMI-DEV model. These improvements are being developed in the form of model redlines. These redlines were reviewed by the high maturity lead appraisers and the CMMI Steering Group at a workshop in late September. As a result of the review, the Steering Group determined that modernizing the practices for levels 4 and 5 was a better choice than a more limited attempt to clarify the information. These improvements will be incorporated into the planned release of CMMI, Version 1.3 for all three CMMI constellations rather than in a CMMI-DEV, Version 1.2a model.The Steering Group and the SEI also recognize that suggested improvements to high maturity practices must come from a wider community than only the appraisers. Therefore, more opportunities like the September workshop are being considered. Where possible, we will link with events such as the November CMMI Conference in Denver and the March SEPG Conference

(< 5 minute) survey.

in San Jose by providing opportunities to contribute ideas in special sessions held at these conferences.

Also planned are improvements and clarifications to the auditing processes now conducted by the SEI to evaluate high maturity appraisals. If an audit results in decertification, an appeal process has been defined that will provide a way for the lead appraiser to request a review of the judgment made by the CMMI Appraisal Program. Documentation of the appeal process is being distributed to high maturity lead appraisers for their use. This appeal process includes a review of the contested appraisal by a board of lead appraisers, with the voting members from outside the SEI.

If the result of an audit is that the appraisal results are rejected by the SEI, a process is available to the organization to contest this outcome. For organizations concerned about appraisal results, an adjudication process is being made available to assure the organization's understanding of the reasons for the SEI's determination. Further, the process allows a review by high maturity lead appraisers external to the SEI to provide an independent cross-check of the SEI quality assurance determination. Detailed descriptions of these processes are found on the SEI's Appraiser Program pages. www.sei.cmu.edu/appraisal-program/appraiser-communications/index.html.

Summary

High maturity thinking has come a long way since the levels associated with the term were created in the early 1990s. The effective use of statistical techniques to manage and improve processes has advanced with the continuing evolution of tools that assist in measuring and predicting process performance. The CMMI Steering Group and SEI believe that the steps described in this column will help ensure that CMMI users more clearly understand and interpret CMMI practices and achieve their process improvement goals.

About the Author

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.

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.

Crisis Management

NEWS AT SEI

Author

Watts S. Humphrey

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

Process Improvement

CMMI

This article was originally published in News at SEI on: May 1, 2008

The idea for this column was suggested by Anand Paropkari and, I am ashamed to say, he made that suggestion an embarrassingly long time ago. In any event, his point was a good one. He said: "Most IT managers think that CMM and CMMI are not for them. Their perception is that the models are developed for IT vendors or DoD contractors."

I hear this all too often: "We are different." These folks seem to be saying that what worked for somebody else will not work for them. If we take this argument to its extreme, nobody could ever learn from anyone else, and we would all be stuck in whatever rat trap we happened to fall into. Then we would all have to learn by ourselves how to solve our own problems. If our forebears had all followed that strategy, we would never have progressed out of the stone age. Of course everybody is different as is every organization. But, at least in the software business, we share a great many common problems.

The Key Question

The fundamental question is whether your boss wants to improve the way the work is done. If he or she does, you probably will not have much trouble making the case for exploring an improvement effort. Then the problem would be to figure out how to get the boss' attention, what kind of improvement method to propose, and how to make the case.

More likely, however, your boss' saying "We're different" is just a way of saying "Don't bother me, I'm busy." When your boss is not interested in process improvement, there are two likely reasons. He or she is either a plodder or is too preoccupied with the

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

current crises to think about anything else.

A Plodder Example

By plodder, I don't mean someone who is lazy but rather a manager who is comfortable with the way things currently work and doesn't want to be bothered with some new improvement effort. Such people will usually give an excuse like "we tried that and it didn't work" or cite an example of an improvement effort that failed. A more sophisticated management response is to ask for a return-on-investment analysis. Since such studies typically take a lot of effort and are easy to shoot down, such requests are guaranteed to indefinitely postpone further talk of improvement.

A good example of the plodder case was the director of a laboratory where I worked many years ago. After I had explained my improvement proposal, he asked how long it would take to recoup the investment. I explained that, realistically, it would take from five to 10 years. He then answered: "But I retire in five years." I thanked him for his time, rolled up my presentation flip charts, and started looking for another job. With bosses like this, you only have two choices: outwait them or leave. I left.

The other case is much more likely. It is that your boss is so enmeshed in the current crises that he or she doesn't have the time or energy to think about anything else. This is the most common case, and it is the one most of us need to think about, either as workers trying to convince busy bosses to make changes or as bosses who are too busy to think about anything but surviving. In either case, if you handle the situation properly, you have a good chance of actually making some improvements.

The way to deal with a busy boss is built into the Capability Maturity Model Integration CMMI) model and the improvement methods such as the Personal Software Process (PSP) and Team Software Process (TSP) that grew out of it. The best way to appreciate the CMMI strategy is to realize that the move from Level 1 to Level 2 is actually a move from crisis-driven management to plan-driven management. That is a fundamental change. It will show immediate and lasting benefits and should be top priority for any busy boss.

The Power of Planning

The great power of plan-driven management was driven home to me many years ago. I worked at IBM, and the company had announced and was starting to ship a new line of hardware products. Unfortunately, the software to support these products was late, and the schedule had slipped three times. At that point, the company fired the director of programming and gave me the job. I now had nearly 4,000 programmers working for me in 15 laboratories in the United States and Europe. They were all working to develop parts of this system, and everyone was late. The customers were irate, the marketing force was in turmoil, and everybody demanded delivery dates from me RIGHT NOW! This indeed was a crisis.

When I arrived in my new office the first day, I quickly made two decisions. First, I wanted no mail unless it came from my boss, his boss, the senior VP, or the chairman. My predecessor had spent most of every day and night just reading and answering his 3-foot stack of daily mail, and my large and overworked office staff spent all of its time just producing a 20-page mail summary. Clearly, reading and answering mail had not prevented and would not solve the current crisis.

The second thing I needed to do was to find out what was going on in the development groups. To do this, I scheduled trips to the three largest development laboratories. I found the same story at every one. Nobody had any plans and they didn't even have a prioritized list of what they had to do. Everybody was just banging out and testing code as fast as they could. I then asked the management team at each location how they would do the job if they did it in the best way they knew. They all agreed that they would establish clear priorities lists, nail down the requirements, and develop plans.

When I asked them why they didn't do that, they all said the same thing: "We don't have the time." This was clearly nonsense! The best way to do the job had to be the fastest and cheapest way. These folks, as smart and competent as they were, were clearly in panic mode and flailing.

Crisis Management is an Executive Problem

The problem was that all of these managers had an enormous list of things that they had to do. But to ship code, the only things that were essential were coding and testing. So they spent their time coding and testing, and they were then so busy that they didn't have time to do any of the other things that they knew they "had to do." While everyone believed that planning was important, it wasn't an essential prerequisite to shipping code and therefore was considered optional.

It took me a little while to realize that, even though I had only been in the job for a couple of weeks, I was responsible for this whole crisis. I then went to the senior VP and told him I was going to stop everything. He turned pale. However, after I explained what I planned to do and why, he agreed. Since all that the developers could do was respond to crises, I had to turn planning into a crisis. I then sent a directive to all of the laboratories. Henceforth, no one could ship a program or announce a program, and I would not even fund a program until I had plans on my desk. They had 60 days to produce their plans and to review them with me personally. This made planning a crisis.

At the same time, I gave a talk to each of the four marketing regions to explain to the sales force what we were doing. I said that we had cancelled all of the software schedules and that we were busily developing plans for the work. When we had the plans in 60 days, we would announce new schedules. Furthermore, these would be schedules that they could believe. Several salesmen later told me that, for the first time, they were able to explain what was going on to our customers and that, while the lack of dates was a problem, they could start selling again.

The Plan Reviews

In the plan reviews, I found lots of holes. Some plans weren't synchronized with planned release schedules, others left out quality assurance or documentation, and a few even left out integration and final testing. I didn't cut a single schedule, but I did add a 90-day cushion to all of the release end dates. As I explained to the development managers, they might be embarrassed if they missed a release date; but I could get fired.

While we gradually ate up the 90-day schedule cushion, we did not miss a single delivery date for more than two years. The change in the organization was dramatic. This group that had never before met a schedule was now consistently delivering on or ahead of plan. People now had time to think, product quality improved dramatically, and many of the developers and managers were even getting home for dinner with their families. Now, instead of being bums, the programmers were treated like the true heroes they were.

Getting Your Boss' Attention

Now that you are convinced that crisis management is a black hole, how do you convince your manager? There are three general ways to attack this problem. The first is by direction. To do this, however, you have to go over your boss' head and get higher management to issue a directive. While this will almost certainly work, it is generally very risky unless your father is company president or your uncle is a majority stockholder.

The second approach is through logic. If you work directly for the executive who has the power to resolve this problem, and if you are on good enough terms to have such a discussion, this is probably a good next step.

The third approach, however, is almost always effective. That is to identify those elements of the recurring crises that are under your personal control. Then figure out how to establish plans to resolve the crises before they blow up. You might even involve some co-workers in doing this with you and cover as many of the crisis causes as you can identify and control. Then, after you have established and used these plans long enough to demonstrate their effectiveness, go to your boss and explain what you did and why it worked. Assuming that your boss finds what you did helpful, that would provide the opening for broadening your efforts to include more of your boss' territory and, ultimately, even to start looking at a more comprehensive improvement strategy.

Getting Started

After getting your boss' attention, you need to have a concrete plan of action. The key to any improvement plan is to start with a modest initial effort but to also ensure that this effort will produce tangible positive results. For example, you could either look at a small part of CMMI such as product planning or start with two or three TSP projects [Chrissis 07, Humphrey 02]. Regardless of the approach you take, it is important to have management's agreement.

While you can often accomplish a lot by yourself, any CMMI improvement effort must have reasonably broad coverage to make a measurable improvement in the organization's performance. For broad coverage, you need senior management support. Similarly, while initial TSP introduction can generally be confined to a small part of the organization, you will need senior management support here as well. This is both because the professional guidance to launch even a small TSP effort costs money and also because TSP introduction calls for a change in management style. However, management style is set by senior managers, so they must be involved.

A Final Caveat

All process improvement programs generally encounter a hidden trap. The problem is that crisis management is an all-consuming activity. Everybody is working hard, people are staying late, a lot is going on, and there is a great feeling of accomplishment. This is the typical reaction of people who confuse speed with progress. Highly professional and efficient work, however, is almost invisible. The product just gets delivered on schedule and works. What is the big deal?

What this means is that, after you take the first improvement step and it is successful, most of the causes for your crises will now be prevented and the immediate rationale for the improvement effort will have gone. If you haven't built the case for the next improvement step before you reach this point, the entire improvement effort will almost certainly be truncated and you will be back where you were at the beginning.

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, Harry Levinson, and Bill Peterson.

References

[Chrissis 07]
Chrissis, Mary Beth; Konrad, Mike; & Shrum, Sandy. *CMMI: Guidelines for Process Integration and Process Improvement. 2nd ed*. Reading, MA: Addison Wesley, 2007.

[Humphrey 02]
Humphrey, W.S. *Winning with Software: an Executive Strategy*. Reading, MA: Addison-Wesley, 2002.

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 11 books. His most recent books are *Winning with Software: An Executive Strategy* (2002), *PSP: A Self-Improvement Process for Software Engineers* (2005), *TSP, Leading a Development Team* (2006), and *TSP: Coaching Development Teams* (2006). 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. In a White House ceremony in 2005, President George W. Bush awarded him the National Medal of Technology.

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

The ROI of Security

NEWS AT SEI

This article was originally published in News at SEI on: May 1, 2006

Return on investment (ROI) is the heart of business—it's why we form companies, develop them, merge them, and even sell or close them. But, as applied to information security, determining ROI has always been a challenge. How do you quantify the value of something that is *less likely to* happen if you spend money to prevent it?

Increasingly, however, ROI is a necessary metric for evaluating security investments. If you're a chief information officer (CIO) or chief information security officer (CISO), you may be asked to provide hard numbers rather than relying on fear, uncertainty, and doubt to sell your proposals. If you're a CEO, you may ask for hard numbers rather than simply being willing to approve a security investment "because it must be done, or else." You also may recognize the wisdom of using hard numbers to compare several security-investment alternatives and choose the one that best contributes to the organization's bottom line.

This column explores the importance of ROI metrics to an organization's security efforts and why it's important to formalize these metrics.

ROI as Risk Reduction

First, let's touch briefly on what ROI means. Return on investment measures the expected improvement over the status quo (for example, a 50 percent reduction in the number of firewall breaches) against the cost of the action required to achieve the improvement (for example, the purchase of a new firewall).

Note that in the context of security, improvement is generally characterized not as a concrete gain, but rather as a *reduction in risk*.

So, how do you determine how much you'll save if you make a specific investment? There are several methods you can use, but first you need to decide what to measure. Here are six types of loss you might consider for your ROI calculations [Allen 03]:

1. **Lost productivity**. How many employees would be unable to get work done because of a security breach, and for how long? What if their computing equipment were seized by law enforcement for forensic analysis? How much time would be spent by IT staff repairing damage caused by the breach as opposed to

doing other work?

2. **Loss of revenue during outages**. Do you operate a web site that could be taken down by a security incident? How much revenue might you lose per minute, per hour, or per day in this scenario? What if you lost Internet connectivity?
3. **Loss of data, temporary or permanent**. Restoring from a backup can be costly. If an insider destroyed your backups and then deleted data, it could be catastrophic.
4. **Compromise of data through disclosure or modification**. Strategic and product plans, as well as sensitive financial information, are just a couple of examples.
5. **Repair costs**. You might need to buy new hardware or use disk-recovery services, for example.
6. **Loss of reputation**. Think about how you'd feel reading about the breach on the front page of a newspaper. More than 20 U.S. states, most notably California, have or are considering laws that require disclosure of incidents that compromise customer data [Rasch 05<; PIRG 06]. Do you do business in these states?

This is only a partial list. Think about other areas you could measure. Do you take into account the indirect costs of a breach to your customers, suppliers, and stakeholders? How can you really calculate that?

As you might suspect, there's a degree of uncertainty and subjectivity involved, but once you decide what to measure and estimate, the question of how to measure it should be somewhat easier. The most effective measures are likely to be those you already are using, because these will enable you to compare security projects with all other projects. Security is only one aspect of business, albeit an important one; as such it's best to analyze it in concert with all other investments and expenses.

Payback

One typical method for measuring security ROI is *payback*, a simple calculation that compares annualized loss expectancy against the expected savings as a result of an investment.

Here's an example: You determine that your chance of suffering a breach due to password compromise is approximately 90 percent in any given year (based on historical incidents and surveys). You determine that if you institute a security-awareness program, you can increase the use of strong passwords throughout the organization and reduce your chance of a password compromise incident to approximately 30 percent in any given year.

If password compromise costs your organization, on average, $150,000 per incident, and you've lowered the chance of such an incident taking place from 90 percent to 30 percent, you will have reduced your total annualized loss expectancy from 90 percent of $150,000 to 30 percent of $150,000 [Allen 03]. Your expected loss will have dropped from $135,000 per year to $45,000 per year.

Have you saved $90,000 per year? No. If a password breach still occurs, you've saved nothing in that year. But breaches will happen less often. The annualized loss expectancy helps you calculate the expected magnitude of savings from reduced risk, so *on average* you will have saved $90,000 per year by instituting the training program.

In this example, your decision about whether or not to invest could change based on the cost of the training program. For example, if the program cost only $10,000 per year to implement and maintain, the clear decision would be to make the investment. Conversely, if the program cost $100,000 per year, it clearly would not be worthwhile. But what if the program cost $80,000 per year? Then the decision would be more difficult, as we'll see later in this column.

Net Present Value (NPV)

A second method, *NPV* or *net present value*, adds another dimension to payback by considering the time value of money–the fact that money spent today is worth more

than savings realized tomorrow. The best thing about NPV as it relates to security is that many other units of the organization–such as Finance–are probably using it.

For the calculation we just did for the security-awareness program and the reduced password compromises, using NPV would change the exact numbers in the result significantly, though it would still be obvious that the awareness program shows a strongly positive ROI.

For example, if you were using NPV and looking at money you'd be saving today, nothing would change. However, with security-awareness training, the savings are in the future. Each year, you would expect to save $90,000. However, $90,000 tomorrow is worth less than $90,000 today.

NPV does not just apply to future savings. Future costs also are cheaper than costs incurred in the present. For example, a firewall that requires a $100,000 payment up front actually costs more than a firewall that requires a $100,000 payment two years from now.

So, if you're calculating projected savings or costs using NPV, you need to know the discount rate, which determines how much *less* your money will be worth in the future. This isn't something your organization can determine; it's a rate that affects the economy as a whole at any given time.

Let's say the discount rate is 10 percent. If you expect to save $90,000 a year from now, NPV will tell you the value of that savings *in today's dollars*. To calculate NPV, divide the expected yearly savings ($90,000) by 1.1 (that's 1 plus the discount rate) to the power of 1 (because we're calculating one year out). The result is approximately $82,000.

Carrying our example further, the $90,000 we save *two* years from now will be worth $90,000 divided by 1.1 to the power of 2 (because now we're calculating two years out). The result is approximately $74,000. For year three, it's $90,000 divided by 1.1 to the power of 3, for a result of about $68,000.

If you're trying to weigh costs and benefits, and some of the costs are immediate but the benefits are long term, NPV can provide a more accurate measure of whether a project is truly worthwhile.

The Role of Risk Management

It is important to note that, as in any risk assessment scenario, the numbers derived from such calculations are not precise and should not be viewed as such.

For example, you have to determine the *cost* of a breach to your organization. The cost of a password compromise to a financial-services firm won't be the same as the cost to a clothing retailer. You have to look at your business, assess your risk and historical losses from categories of security breaches, and determine a number that represents an accurate estimate of these costs per incident. This, of course, implies that you're collecting and tracking these costs.

Likewise, historical data can help you determine your *risk* for each kind of incident. For example, what is the probability that your organization will experience an insider attack? How about an earthquake that destroys hardware and severs network connectivity?

These numbers will be unique to your organization. If you're a high-profile target, you may discover that your risk of a certain kind of break-in is 100 percent in any given year, and you'll probably be willing to spend more to mitigate it than an organization that faces just a 20 percent risk.

Lastly, you need to determine how much you think a security investment is going to *lower* your risk.

As a result, there's always going to be some subjectivity in the numbers you use to calculate the value or benefit of security investments. Even so, these numbers have value. The value is in their consistent use across multiple projects. The same ROI

method applied consistently across all projects will produce a meaningful measure of value and a means for comparing among alternatives.

For example, if you evaluate a new firewall purchase in the same way that you evaluate a security-awareness training program, you can make a rational judgment about which investment will produce a greater ROI.

Unfortunately, ROI has rarely been measured consistently for security projects in the past, if it was measured at all. Many security proposals are approved based on fears of public disclosure of customer information, regulatory compliance, or pressure to keep pace with competitors. That doesn't mean they aren't necessary, but it does make it difficult to gauge their worth in terms of ROI and to compare proposals against one another.

In summary, when determining the expected costs and benefits of various security investments, consistency is key, as is the quality of your risk assessment.

Assess Early and Often

The timing of your ROI assessment is also important. It's best to calculate returns before any capital is spent, although an ROI evaluation of ongoing projects could help you determine how they are progressing in comparison with other projects. In fact, it's probably advisable to conduct follow-up assessments to make sure you're getting the desired return on investment. But these follow-up reviews shouldn't take the place of an initial, thorough comparison of alternative investments.

You may have given equal weight to a new firewall purchase and a training program, for example. Subsequently, you may have found that the training program produced significant positive returns while the firewall purchase did not. Some sort of prioritization should have occurred before these investments were made, so that the training program took precedence over the firewall. Did it?

Without a consistent metric, the answer could be "maybe or maybe not." And that's the antithesis of ROI. ROI gives us criteria for making better-informed decisions and choices.

Not the Same ROI

This discussion may have made it sound as if security ROI is just like ROI for other projects. In some ways, that's true, but in other ways, it isn't. Economists who've studied security ROI have discovered that straight numbers don't tell the whole story. According to economists Lawrence Gordon and Martin Loeb of the University of Maryland, you should never spend more than 37 percent—and often less—of your expected savings on a security investment [Gordon 02]. Above that level, it usually doesn't pay off in the real world. There are several possible reasons for this, including the high level of uncertainty inherent in the calculations.

So, for example, if the NPV of a security-awareness training program is $200,000, you probably shouldn't spend more than $74,000 in today's dollars on the program.

Remember, though, that in the context of a whole business with many security projects competing for limited funds, the most important thing is not the hard numbers. The most important thing is that you use the same defined set of criteria to evaluate each potential security project. For example, it is misleading to compare an ROI calculation for a firewall purchase that does *not* take into account the potential for loss of reputation with an ROI calculation for an intrusion-detection system purchase that *does* take into account this potential loss. Even if you use NPV for both calculations, you are not measuring the same thing.

Likewise, it is misleading to compare a payback calculation for a firewall with an NPV calculation for an intrusion-detection system, even if the exact same criteria are taken into account. In this case, the inconsistency in methods results in a meaningless comparison.

This is not about hard numbers. That's not where the value is. The value is in the ability to compare projects and choose wisely among them. Consistency is what is most important, both in what you choose to measure and in how you choose to measure it.

Make your decision, and then stick with it.

The Point of Policy

Once you decide what to measure and which method to use, write it into your security policy and ensure that everyone reports out accordingly. It's vital to maintain consistency over time. If you've been using NPV to evaluate security projects and you start using payback—*especially* if other divisions in your organization are still using NPV—you're going to find it hard to compare new projects with ongoing ones.

Policy establishes stable guidelines. Once something is in policy, it's harder to change it. This should hold true for ROI as much as for other aspects of security. So first, decide what you're going to measure; second, decide how you're going to measure it; and third, write it into your security policy.

References

[Allen 03]
Allen, Julia. "Making the Business Case for Information Security: Selling to Senior Management." Carnegie Mellon University at InfoSec World 2003, March 10, 2003.

[Gordon 02]
Gordon, Lawrence A., and Martin P. Loeb. "The Economics of Information Security Investment." *ACM Transactions on Information and System Security, Vol. 5*, No. 4, November 2002, p. 440.

[PIRG 06]
http://www.pirg.org/consumer/credit/statelaws.htm

[Rasch 05]
Rasch, Mark. "Cleaning Up Disclosure." *SecurityFocus*, April 11, 2005.

About the Author

Stephanie Losi is a graduate student in the Information Security Policy and Management Program at Carnegie Mellon University. She is working with CERT to develop security awareness content for executives and information security personnel. In addition to her work at CERT, Losi has authored online courses dealing with business ethics and served as managing editor of the *E-Commerce Times*. She earned her bachelor's degree in journalism from Northwestern University.

Understanding Architectural Patterns in Terms of Tactics and Models

NEWS AT SEI

Authors

Felix Bachmann

Len Bass

Robert Nord

This article was originally published in News at SEI on: August 1, 2007

Architects use architectural patterns and tactics to improve software architectural decisions and simplify the design process. In this column, we explore the relationships of tactics and architectural patterns, focusing on the quality attribute of modifiability.

Patterns are solutions that resolve multiple forces, whereas tactics focus on specific quality attributes. To more effectively apply both tactics and patterns, architects need to understand how architectural tactics and patterns relate and how to use them effectively.

The attribute-driven design (ADD) method [Wojcik 2006] provides guidance on when to apply patterns and tactics while designing the architecture but leaves the ordering to the discretion of the designer. Architects are advised to choose a pattern if they can find one and then adjust the pattern based on tactics. They are advised to use tactics when they need help coming up with a pattern, when an existing pattern isn't quite right and they need to tailor it, or when they want to validate the choice of a pattern. To help with this activity, Table 1 shows the relationship of some well known architectural patterns [Buschmann 2007] to their corresponding tactics that address modifiability.

Architectural Patterns	Tactics for Modifiability									
	Increase Cohesion		Reduce coupling					Defer Binding Time		
	Maintain Semantic Coherence	Abstract Common Services	Use Encapsulation	Use a Wrapper	Restrict Comm. Paths	Use an Intermediary	Raise the Abstraction Level	Use Runtime Registration	Use Start-Up-Time Binding	Use Runtime Binding
Layers	X	X	X		X	X	X			
Pipes and Filters	X		X		X	X			X	
Blackboard	X	X			X	X	X	X		X
Broker	X	X	X		X	X	X	X		
Model View Controller	X		X			X				X
Presentation Abstraction Control	X		X			X	X			
Microkernel	X	X	X		X	X				
Reflection	X		X							

Table 1: Architectural Patterns and Corresponding Tactics for Modifiability

Modifiability tactics are based on analytic models [Bachmann 2007]. They can be thought of as the transformations that are available to affect the behavior of a design with respect to modifiability. They provide the architect with a view of the analytic model in terms of a limited number of parameters based on coupling and cohesion, and on cost models. Figure 1 shows tactics that support modifiability and are organized based on these models to increase cohesion by localizing changes and reduce coupling by preventing ripple effects. Furthermore, the coupling and cohesion-based tactics enable the deferred binding time tactics.

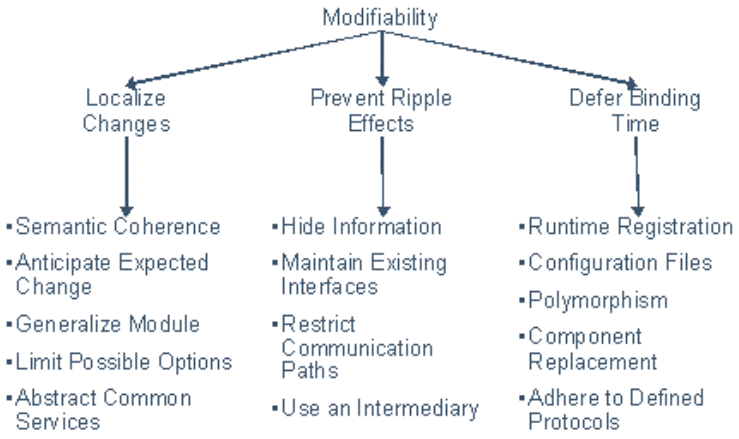

Figure 1: Tactics to support modifiability [Bass 2003]

Consider an example where a change request is made that specifies that the modification is made with no side effects in three hours. To understand the implications of the proposed change, the architect would interpret the architecture of the system to create a model for reasoning about modifiability. One such interpretation is shown in Figure 2.

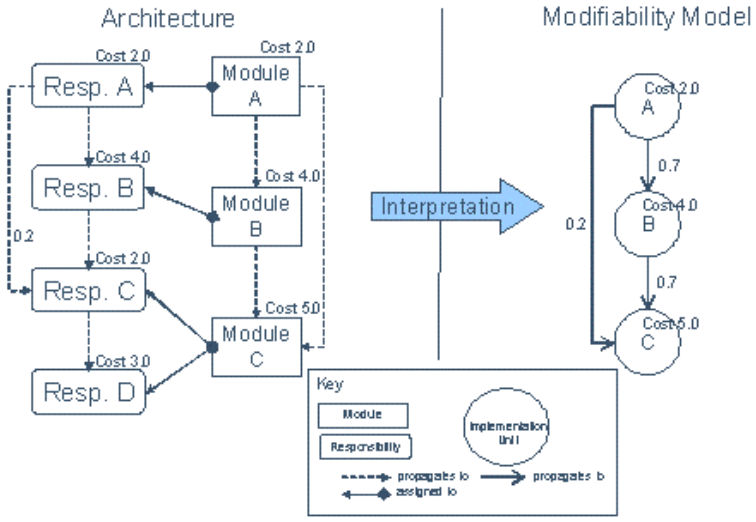

Figure 2: Using the architecture to reason about modifiability

The architecture is defined in terms of modules and their associated responsibilities. Costs are associated with the responsibilities and the dependencies between them represent the probability that a change to one responsibility will propagate to another. The corresponding modifiability model shows the associated costs and probabilities of propagations for the modules, based on the responsibilities that are assigned to them.

Consider the case when the proposed change affects module A. The modifiability model shows that the cost of changing module A is 2 hours. Furthermore, there is a 0.7 probability that the change to module A will propagate and require a change to module B and a 0.2 probability that the change to module A will propagate and require a change to module C. The model predicts that the change will take:

$$2.0 + (0.7 * 4.0) + (0.2 * 5.0) = 5.8 \text{ hours}$$

This predicted response is compared with the three hour desired response and found to be lacking. The architect has the option of making changes at the tactic level or the pattern level:

1. apply tactics – the model indicates that there is a problematic dependency chain between modules A and B which contributes 2.8 hours to the predicted time for modifying the system. The architect can minimize the ripple by applying tactics for preventing ripple effects such as the *use an intermediary* tactic to decouple responsibilities in the architecture and the *restrict communication paths* tactic to prevent modules from bypassing the intermediary.
2. apply a pattern – the *layered* pattern can be used to break the dependency between modules A and B. To validate the choice of the pattern, it can be understood in terms of the *use an intermediary* and other tactics shown in Table 1 and reasoned about in terms of the associated modifiability model.

Patterns have widespread use in the design process. By relating patterns to tactics, practitioners gain a basis for making principled design decisions. The designer can use tactics to understand patterns and how they support quality attributes such as modifiability, or to adjust patterns in order to improve their properties.

Modifiability is one important quality, but it is not the only one. We anticipate providing similar analysis for other important quality attributes.

References

[Bachmann 2007]
Bachmann, F.; Bass, L.; and Nord, R. *Modifiability Tactics,* (CMU/SEI-2007-TR-002). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2007.

[Bass 2003]

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

[Buschmann 1996]
Buschmann, F.; Meunier, R.; Rohnert, H.; Sommerlad, P.; & Stal, M. *Pattern-Oriented Software Architecture: A System of Patterns*. Chichester, NY: Wiley, 1996 (ISBN: 978-0-471-95869-7).

[Wojcik 2006]
Wojcik, R.; Bachmann, F.; Bass, L.; Clements, P.; Merson, P.; Nord, R.; Wood, B. *Attribute-Driven Design (ADD), Version 2.0*, (CMU/SEI-2006-TR-023). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2006.

About the Authors

Robert L. 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 internationally on architecture-centric approaches. He is co-author of *Applied Software Architecture* (2000) and *Documenting Software Architectures: Views and Beyond* (2002).

Len Bass is a senior member of the technical staff at the Software Engineering Institute who participates in the High Dependability Computing Program. He has written two award-winning books on 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.

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.

Spider-Man 3 Developers to Discuss Process-Improvement Experience in SEPG Keynote

NEWS AT SEI

Author

Heidi Brayer

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

Process Improvement

This article was originally published in News at SEI on: August 1, 2007

As Guha Bala tells it, everything could have stopped after Synnergist, the first game released by Vicarious Visions, a computer and video game company he and his older brother, Karthik, formed while still in high school.

“We thought it would be easy. We found out it was much more complicated than we ever anticipated,” Guha says of their initial venture into the gaming world.

The two brothers could have pursued other careers and just chalked it up to an adult-world learning experience from a hobby started in the basement of their parents’ home.

After all, each had other interests, the kind of interests that equate to job security. Guha would go on to earn an honors degree in chemistry from Harvard University and was considering whether to become a scientist or pursue some other graduate-level course of study. Karthik, meanwhile, was studying at Rensselaer Polytechnic Institute.

But both brothers chose to develop software titles and now lead Vicarious Visions, which recently topped $1 billion in retail sales with more than 100 software titles.

“I found this is what I was passionate about, getting a lot of energy with very little promise of anything in return,” said Guha during a recent telephone interview. Guha and Karthik Bala will keynote the SEPG North America conference held March 17 to 20 in Tampa, Fla.

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

Their talk will draw upon their experiences in process improvement developing more than 100 software titles including *Bee Movie*, *Transformers*, *Shrek the Third*, *Tony Hawk* and in May, 2007, *Spider-Man 3* for five different gaming platforms including PlayStation 2, Nintendo Wii, Nintendo DS, and Nintendo Gameboy Advance.

For *Spider-Man 3*, the systems and sub-systems code alone amounted to approximately 1 million lines of code.

Guha, who serves as president of Vicarious Visions and oversees the day-to-day production management, explained that developing computer and video games like *Spider-Man 3*, involves the coordination of numerous outside influences.

The team that worked on the project had to coordinate the efforts of nearly 200 people within 16 months to meet production schedules. They had to oversee such tasks as scheduling recording time with actors Kirsten Dunst and Tobey Maguire. Other aspects of the project including writing a script that, in its final form totaled 6,000 lines of dialogue, which had to complement the movie screenplay, but not duplicate it. Another aspect was the sound technology which includes writing audio code.

“There’s a sound engineering process for taking a voice recorded in a studio and making it sound like it’s recorded in a warehouse,” Guha explained.

To coordinate such an extensive undertaking, Guha said employees at Vicarious Visions often incorporate components of the Team Software Process (TSP) and Personal Software Process (PSP) methodologies into their work. The two methodologies were created by Watts Humphrey, National Medal of Technology winner and founder of the SEI Software Process Program, as a way for development engineers and teams to implement principles of CMMI in their organizations.

For example, before even beginning a project, individuals and teams at Vicarious Visions rely upon historical data when making future estimates on project schedules. Another important aspect has been the implementation of a team-wide launch process when beginning new initiatives.

“This involves breaking down the elements of a process and building a shared vision and clarity of what we’re actually trying to accomplish,” he said, adding that it is a way to bring the team members’ expectations into alignment with management and also build broad-based commitment to a project.

Guha said it was while he was in college that he and Karthik, who was attending RPI at the time, decided to make the uncertain world of gaming their future.

At RPI, Karthik had met and was inspired by Mike Marvin, who, with four other students, started a software company called MapInfo Corp., which was later sold to Pitney Bowes. Marvin, who received his Master of Science degree from the Graduate School of Industrial Administration at Carnegie Mellon University, mentored many tech startups at the time including Vicarious Visions.

"Mike Marvin said he'd give us seed capital if we stayed here, and persuaded us that we could learn plenty from the other companies," Karthik stated in a Dec. 19, 1999, *New York Times* article about software startups in central eastern New York. The two brothers had been considering moving their company to California after graduation in 1997. They are now based in Albany and employ more than 150 people. Marvin and others helped Vicarious Visions with a business plan including benefits for its game developers. Other software executives in the region also helped in developing strategies for syndicating games on the World Wide Web, according to the *Times* article.

Gutha said he and his brother have no regrets and their new game plan involves looking to the future.

“The simplest way to explain it is this: Our goal is to make Vicarious Visions a household name for our medium,” Guha said.

Share This Page

For more information

Contact Us

info@sei.cmu.edu

412-268-5800