Software Engineering Institute

Proceedings of the Second Software Architecture Technology User Network (SATURN) Workshop

Robert L. Nord

August 2006

TECHNICAL REPORT
CMU/SEI-2006-TR-010
ESC-TR-2006-010

Software Architecture Technology Initiative

Carnegie Mellon

This report was prepared for the

SEI Administrative Agent
ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2100

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about purchasing paper copies of SEI reports, please visit the publications portion of our Web site (http://www.sei.cmu.edu/publications/pubweb.html).

Table of Contents

List of Figures

List of Tables

Acknowledgements

Thanks to Linda Northrop for sponsoring the Software Architecture Technology User Network (SATURN), to Paul Clements for co-organizing SATURN, to the members of the Software Architecture Technology (SAT) Initiative at the Carnegie Mellon® Software Engineering Institute (SEI), and to the SEI for supporting the workshop. Laura Huber and Pat McDonald provided administrative support. Ruth Gregg and Linda Canon handled the local hotel arrangements. Eileen Forrester, Gian Wemyss, David White, and David Zubrow provided feedback on designing the working sessions and provided facilitation support. The working sessions were made possible by Felix Bachmann, Len Bass, Matthew Bass, Mike Gagliardi, Rick Kazman, and Rob Wojcik, who prepared the agendas for the sessions, moderated discussion, and wrote the summaries that appear in this report. Amine Chigani took notes during the plenary sessions. Phil Bianco, Larry Jones, Bob Krut, Paulo Merson, and Ipek Ozkaya were scribes for the working sessions. Bob Krut designed and maintained the SATURN Web site. Bob Fantazier provided graphic arts support and Pennie Walters provided technical editing support.

Thanks to the SATURN Steering Committee for nurturing the growing SATURN idea and getting involved in the planning details to make the workshop happen: Nanette Brown, director, Applied Architecture and Quality Assurance, Pitney Bowes; Linda Northrop, director, Product Line Systems Program, SEI; and Rolf Siegers, chief architect and engineering fellow, Raytheon.

Thanks to all who participated by presenting, working in the breakout sessions, and engaging in the discussions. Those activities all contributed to what people say they liked best about the workshop: “quality content, diverse presenters, interactive, flexible, and informal-feeling sessions,” “open atmosphere for discussion,” and “collaboration and sharing of ideas.”

SATURN Workshop Series

The goal of the Software Architecture Technology User Network (SATURN) workshop series is to bring together software systems engineers, architects, technical managers, product managers, and researchers to share experiences using software architecture technology developed by the Carnegie Mellon® Software Engineering Institute (SEI). Participants discuss ideas, issues, and needs related to software architecture practices. They also develop a network of individuals who are interested in using and improving those practices. SEI architecture-centric methods include the SEI Quality Attribute Workshop (QAW) [Barbacci 03], the SEI Attribute-Driven Design (ADD) method [Bass 03], SEI Active Reviews for Intermediate Designs (ARID) [Clements 02], the SEI Architecture Tradeoff Analysis Method® (ATAM®) [Clements 02], the SEI Cost Benefit Analysis Method (CBAM) [Bass 03], the SEI Views and Beyond (V&B) approach to documentation [Clements 03], and the SEI Architecture Reconstruction and Mining (ARMIN) tool [Kazman 02]. These methods are based on a core set of attribute models, reasoning frameworks, and architectural tactics.

Participants in the workshop discuss the challenges they face in meeting quality attribute requirements, predicting quality attribute behavior, and making practical and informed tradeoffs about quality attributes early in the software development life cycle. SATURN provides a unique opportunity to learn from fellow participants about how to use effective software architecture practices throughout the life cycle to ensure predictable product qualities, costs, and schedules. It also provides an opportunity to give feedback to the SEI about promising future directions in software architecture technology and practices.

® Carnegie Mellon, Architecture Tradeoff Analysis Method, and ATAM are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

Abstract

The second Carnegie Mellon® Software Engineering Institute (SEI) Software Architecture Technology User Network (SATURN) Workshop was held April 25-26, 2006 in Pittsburgh, Pennsylvania. A total of 61 software systems engineers, architects, technical managers, product managers, and researchers exchanged best practices and lessons learned in applying SEI software architecture technology in an architecture-driven development or acquisition project. In the closing session, workshop participants noted these highlights: presentations showing the methods in action, a comparison of multiple SEI Architecture Tradeoff Analysis Method® (ATAM®) evaluations and cross-wise analysis, the workshop format using interactive presentations, a good mix of academic and industry perspectives, and a sharing of workshop results.

This report describes the workshop format, discussion, and results, as well as plans for future SATURN workshops. Key topics covered in the workshop and noted by the participants were the future plans of the SEI's Software Architecture Technology Initiative, the overall integration of software architecture methods and techniques, and the experiences others shared in applying the methods and transitioning them for use. Slides for the presentations and recordings of the keynote talks are available at the SATURN workshop Web site:
http://www.sei.cmu.edu/architecture/saturn/.

1 Introduction

The second Carnegie Mellon® Software Engineering Institute (SEI) Software Architecture Technology User Network (SATURN) Workshop was held April 25-26, 2006 at the Sheraton Station Square Hotel in Pittsburgh, Pennsylvania.

During this workshop, 61 participants exchanged best practices and lessons learned in applying SEI software architecture technology in an architecture-driven development or acquisition project. The workshop consisted of 10 talks (including keynotes), 6 working sessions, a reception, and opening and closing sessions. Mark Klein, technical lead of the SEI's Software Architecture Technology (SAT) Initiative, gave the opening presentation. Linda Northrop, director of the SEI's Product Line Systems Program, led the closing session. Keynote speakers included

- Don O'Connell, software/systems architect, The Boeing Company
- Rolf Siegers, engineering fellow and chief architect of the Garland Engineering Center in Intelligence and Information Systems, Raytheon

Workshop activities spanned the SEI technology transition spectrum from creating usable technologies to applying them to real-world problems and accelerating adoption [SEI 06b]. Working sessions included topics from new SEI research initiatives in architectural competency, bridging system and software architecture, and architecture evolution. Presentations showed how participants are applying the methods and emphasized combinations of methods and techniques in a broader context of software development life-cycle practices and technologies.

This report describes the workshop format, discussion, and results, as well as plans for future SATURN workshops. It is organized as follows:

- Section 2 lists the demographics of the workshop participants.
- Section 3 provides an overview of the presentations.
- Section 4 includes discussions from the six working sessions:
  a. Architectural Competency
  b. Bridging System and Software Architecture
  c. Architecture Evolution
  d. Global Software Development

e. Strategic Risk Management for Architectures

f. Building a Software Architecture Community

- Section 5 includes the closing session discussion that focused on the top ideas emerging from the workshop, workshop highlights, and what to do next.
- Section 6 forecasts the future of SATURN workshops.

2 Participants

The 61 SATURN workshop participants were from the sectors shown in Table 1.

Table 1: Demographics of SATURN Participants

Sector	Number
U.S. Department of Defense (DoD)	3
U.S. DoD contractor	8
U.S. commercial	19
International commercial	3
Academia	5
Federally funded research and development centers (FFRDCs) other than the SEI	1
SEI (staff within the Product Line Systems Program)	15
SEI (staff outside of the Product Line Systems Program)	7

.

3 Presentations

Mark Klein, technical lead of the SEI's SAT Initiative, gave the opening presentation: *Future Directions of the Software Architecture Technology Initiative*. Keynote speakers talked about the process of institutionalizing software architecture within a company and the lessons learned in applying the SEI Architecture Tradeoff Analysis Method® (ATAM®). Presentations showed how participants are applying various SAT methods. Three of the presentations showed combinations of the SEI Quality Attribute Workshop (QAW), the SEI Attribute-Driven Design (ADD) method, and/or the ATAM in a broader context of software development life-cycle practices and technologies. Two of the presentations focused on requirements from an architecture perspective. One half day was devoted to the ATAM. The ATAM's impact is being amplified as external lead evaluators become certified and report results, as Don O'Connell did at this year's workshop and Stephan Ferber did at last year's [SEI 05]. Len Bass presented an analysis of ATAM evaluations conducted by the SEI to find patterns in the risk themes and encouraged external lead evaluators to analyze their data and share their results. Craig Martin presented a proposal for tool support.

The following sections include descriptions of each presentation including an abstract (written by the speaker and edited slightly by SEI staff) and notes taken during both the presentation and subsequent discussion. Presentation slides and audio recordings of the keynote speakers' remarks are available at the SATURN workshop Web site [SEI 06a].

3.1 SATURN OPENING PRESENTATION: FUTURE DIRECTIONS OF THE SOFTWARE ARCHITECTURE TECHNOLOGY INITIATIVE

Mark Klein, senior member of the technical staff, SEI

Abstract

The SAT Initiative at the SEI creates, harnesses, and applies innovations that are then codified as effective software architecture practices and used throughout the development life cycle. Our work is guided by responding to real-world needs, maximizing impact, and basing techniques and methods on theoretically sound foundations. This talk briefly reviews the "state of the SAT Initiative" and then outlines our future research directions.

® Architecture Tradeoff Analysis Method and ATAM are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

Notes

Mark Klein, the technical lead of the SAT Initiative, defined the initiative's focus as ensuring that business and mission goals are predictably achieved by using effective software architecture practices throughout the development life cycle. Software architecture is an integral aspect of achieving software quality attributes. To realize the full benefit of software architecture, organizations must be trained to develop sound architectures for their line of business, whether it is military, commercial, artificial-intelligence focused, or service oriented.

Klein talked about the ATAM as a first step and how it is designed for architectural evaluation against quality attribute requirements. When the SAT Initiative started, it focused on architecture tradeoff analysis but later broadened to everything related to software architecture. He argued that the ATAM is domain independent.

Klein also emphasized the importance of initiatives such as SATURN in driving the push towards new techniques that make software architecture practice more structured and therefore more valuable and effective. He mentioned several work areas and products of the SEI and the status of each. Then, he moved to future directions of the SAT Initiative, namely

- architecture-centric, life-cycle practices (i.e., looking at the other software development phases and investigating how they affect or are affected by software architecture)
- the relationship between software and system architecture practices
- architectural competency

3.2 KEYNOTE SPEAKERS

3.2.1 Architecture Analysis Overview and Observations

Don O'Connell, software/systems architect, The Boeing Company

Abstract

This presentation lists some of the key software/system architectural analysis that Boeing uses on its products. Next, the presentation focuses on the application of the ATAM and QAW to a variety of products over the last three years. Only publicly releasable information is described. An overview of our ATAM process adjustments and focuses are described. Keys to success are highlighted. An overview of ATAM/QAW results and risk mitigation activities are also described.

Speaker Bio

Don O'Connell has worked as a software/systems engineer for the Boeing Company for 23 years and holds a master's degree in software engineering. For the last seven years, Don has worked in Boeing Phantom Works, handling approximately 35 projects as software/system architecture consultant and/or problem solver. The projects span all business units of Boeing. Customers include the U.S. Army, Navy,

Air Force, Marines, Department of Defense (DoD), National Aeronautics and Space Administration (NASA), and Department of Energy (DoE), the airlines of several countries' militaries, and several commercial airlines. Don has worked on commercial airplanes; satellite systems; spacecraft; helicopters; jet fighters; and command, control, communications, computer, intelligence, surveillance, and reconnaissance (C4ISR), ground, sea, and airborne systems—just to name a few.

Notes

Don O'Connell's presentation focused on Boeing's experience in applying the ATAM to evaluate its projects. He said that the ATAM had become an integral part of Boeing's suite of tools used to assess product quality. First, he described where the ATAM fits within the company's tools. Next, he explained the four phases of the ATAM and how he conducted those phases in many Boeing projects.

During an ATAM evaluation, O'Connell helps team members keep track of their current stage in the ATAM process with a poster that illustrates the conceptual flow of the ATAM's phases. He argued that the focus during an ATAM process was on identifying risks rather than issues because architects can control risks but not necessarily issues. In addition, O'Connell discussed how Boeing had used an ATAM approach for projects that are either too small or too large for its standard process. For small projects, Boeing used a one-day ATAM approach. For large projects, the company evaluated the architecture of the top-level project using the ATAM and evaluated modules (subsystems) as they are organized within each subset of the project using additional ATAM evaluations. He observed that Boeing identified risks on large-scale, software-intensive systems. O'Connell described how his team used recurrent risk themes for process improvement by using new evaluation and risk mitigation techniques such as the ATAM. During the evaluations, he noticed that his team lacked an emphasis on testability and therefore needed to add a testing engineer.

O'Connell concluded his presentation with these observations:

- Participating in an ATAM evaluation allows stakeholders to know what is going on and to appreciate how some decisions were made.
- For large projects, it is good to conduct one ATAM evaluation for the overall system and take a one-day, ATAM-based approach for system modules.
- When you have good stakeholder role coverage, you get a good breadth of scenarios.
- The best thing about the QAW is that when you gather the stakeholders in a room and put them on the spot, they are likely to give you the best scenarios possible.
- The QAW produces a snapshot of the quality attribute requirements known at the time of the workshop and documents any unknowns. As these requirements

are refined, there is a problem with finishing the quality attribute requirements document, which continues to evolve throughout the software development process.

3.2.2 Raytheon's Architecture Journey

Rolf Siegers, engineering fellow and chief architect of the Garland Engineering Center in Intelligence and Information Systems, Raytheon

Abstract

Raytheon began an "architecture journey" several years ago to institutionalize architecture as a formal practice throughout the company. Corporate Engineering's senior leadership defined a vision to address a set of software, systems, and enterprise architecture needs across Raytheon's multiple business areas.

Since then, Raytheon has enhanced its architecture competencies through a variety of corporate initiatives including

- establishing a corporate Architecture Review Board (ARB)
- establishing a company-wide, standardized architecting process
- definition and deployment of an architecture training program
- definition and deployment of an internal Raytheon Certified Architect Program (RCAP)
- participation in external architect certification programs
- development of reference architectures
- in-depth analyses of the latest architecture tools
- establishing an architecture repository that is accessible internally
- establishing company-wide, architecture-centric Technical Interest Groups (TIGs)
- collaborations with academic, industry, and government organizations

Speaker Bio

Rolf Siegers is an engineering fellow and chief architect of the Garland Engineering Center in Raytheon's Intelligence and Information Systems. He joined Raytheon in 1984 and leads the corporate Raytheon Enterprise Architecture Process (REAP) Initiative—Raytheon's standardized, company-wide architecture process. Rolf sits on Raytheon's corporate ARB, leading and supporting a variety of architecture-related initiatives.

Rolf's program experience includes leading several multidiscipline software architecture teams for large-scale, software-intensive national and international systems since 1997. He is a certified architect for The Open Group Architecture Framework (TOGAF), Version 8.1, ATAM Evaluator (SEI), and Software Architecture Profes-

sional (SEI). He has previously presented at conferences held by the U.S. DoD, Integrated Defense Architectures, The Open Group, and the International Council of Systems Engineering (INCOSE). Rolf holds bachelor degrees in Computer Science and Mathematics from Huntingdon College and is a member of the Institute of Electrical and Electronics Engineers (IEEE) and INCOSE.

Notes

Rolf Siegers started his talk by describing the kind of projects he has been involved in at Raytheon. These projects were characterized as large-scale, multiyear, multi-million-dollar software-intensive systems. His involvement and experience in these large projects spanned all across Raytheon, which employs about 8,000 people in 18 countries. First, he discussed the business case for architecture in general and software architecture in particular. He cited Barry Boehm's research, which reveals a direct correlation between the amount of architecture effort done in a project and the quality of the system produced. In 2001, government agencies began requesting architecture-centric reviews from their contractors. Another factor that made architecture important is communication. Large projects involve a large number of stakeholders; thus, different views are needed to formally communicate the system's functionality to all stakeholders.

Siegers then moved his discussion to how Raytheon achieved architectural maturity that scaled to other company projects. He emphasized that Raytheon's success in architecture was a result of top management's dedication to and faith in the importance of architecture. Raytheon's first corporate initiative was establishing an ARB to establish a company-wide understanding of architecture. However, the question remained which architecture to emphasize. Based on the size of the company, Raytheon was interested in all three levels of architecture: systems, software, and enterprise. Therefore, Raytheon initiated the Raytheon Enterprise Architecture Process (REAP) project. Its primary focus was on the process of how to architect. Then, other initiatives focused on notation and collaboration among the different architects across the company. Raytheon adopted the ATAM to address not only software architecture issues but also systems and enterprise architectures. They selected the ATAM because the method was domain independent and could be adapted to other architectural domains.

Raytheon's next step was training and certification. Siegers talked in detail about how the company established Raytheon's Certified Architect Program (RCAP) for training and certifying its own architects. In addition, he mentioned that tool support for architecture is still nascent, even though consolidation is occurring due to company mergers and competitor buyouts. He concluded by saying that Raytheon's experience with architecture shows that institutionalizing architecture within a company takes money, time, and talent.

3.3 INTEGRATING METHODS AND TECHNIQUES

3.3.1 Definition and Evaluation of Geographic Information System Architecture Using ADD and the ATAM

Ibrahim Habli, Research Associate, Rolls-Royce Systems and Software Engineering University Technology Centre, University of York

Abstract

The presentation provides an overview of and key findings from the application of the SEI's architectural methods in the definition and assessment of a Geographic Information System (GIS) architecture. This application resulted in the documentation of 22 quality attribute scenarios covering performance, availability, modifiability, security, testability, and usability. Three design iterations were then performed, in accordance with ADD, and an architecture was produced and documented in two architectural views: Module and Component-and-Connector (C&C). A total of 38 distinct architectural design decisions were made; each contributed to the achievement of one or more quality attribute scenarios. Finally, the GIS architecture was evaluated using the ATAM, resulting in the identification of 16 sensitivity points, 10 tradeoff points, and 13 risks that were summarized in 4 risk themes. Lessons learned from applying the SEI's architectural methods revealed that addressing GIS quality attributes systematically at the architectural stage facilitated an unambiguous record of the rationale, assumptions, and dependencies of the critical technical decisions involved in achieving key quality drivers. This, in turn, improved the flexibility, adaptability, and analyzability of the architecture. Additionally, the GIS architectural process proved to be useful for teaching purposes. It is currently used as part of a postgraduate course in software architecture as an example of a systematically defined architecture.

Notes

Ibrahim Habli talked about a case study for a GIS architecture done at the University of York, England. The motivation behind this case study came from both academia and industry. The purpose of this research project was twofold. He wanted first to create an architecture for GIS systems using different architectural techniques and second to create a model to show students how software architecture fits within the overall development life cycle. Habli explained some views of the architecture and discussed the architectural decisions made during the decomposition of the system's functionality. He also explained what each module represented.

The next part of the presentation focused on the lessons learned from this exercise. Habli reported that quality attribute scenarios provided three benefits:

1. understandability. They unambiguously define factors that control the achievement of quality attributes.
2. precision. A response and a response measure offer specific means for assessing GIS architectures.

3. traceability. The decomposition of each quality attribute into scenarios enables the traceability of how an attribute is addressed during architectural design and evaluation.

ADD's benefits include its systematic consideration of quality attributes and a mapping between quality attribute scenarios and architectural decisions. Coupling ADD with the Views and Beyond (V&B) approach to documentation provided well-organized architectural documentation and a record of the architectural design decisions applied, resulting architectural views, and underlying design rationale. The ATAM's benefits include its assessment of the main architectural artifacts (requirements and design), enhancement of architectural documentation, and articulation of the parameters for effective reuse of architectural design decisions.

Habli said that those carrying out the case study did encounter some difficulties applying ADD techniques and performing an ATAM evaluation. Among these problems was the need to have a more accurate prioritization, instead of the basic low-medium-high scale. Also, it was difficult to define a quality attribute response and response measure for some attributes such as flexibility. In addition, at this early stage of design, it was difficult to decompose the system functionality without ignoring some of the main functionality. Finally, Habli argued that the ATAM needs more quantitative evaluation elements added to it. He suggested there is a need for specialized assessment techniques to measure quality attributes such as performance.

In the question and answer (Q&A) session, most questions revolved around the issue of the applicability of ADD techniques and the ATAM in industry. Also, there was a discussion about whether it was appropriate to use techniques such as the ATAM without proper training and certification.

3.3.2 Architecture-Centric Development Method

Anthony J. Lattanze, Associate Teaching Professor, Institute for Software Research International, Carnegie Mellon University

Abstract

Functionality is a measure of how well a system does the work it was intended to do, but functionality is not all that matters in software development. Properties like interoperability, modifiability, and portability matter as much as functionality does. These properties are determined primarily by the software structure—or the software architecture. While many structures can satisfy functionality, few can satisfy it and the other quality attribute properties needed in a system. Achieving quality attributes in a predictable way can be accomplished only by deliberately selecting the appropriate structures early in the development process. This approach is a radical departure from high-speed, lightweight programming methodologies (e.g., extreme programming [XP]) that focus on functionality and prescribe writing software until a product emerges—architectures also emerge in this paradigm. Emergent architectural structures may or may not meet the expectations of the broader

stakeholder community. Unfortunately, architectural shortfalls are not recognized until it is too late and repair is difficult and costly. On the opposite end of the spectrum are methods that espouse high-ceremony processes and a heavy emphasis on document production. While mature processes are certainly beneficial, there is no consistent correlation between high-maturity organizations and high-quality architectures. Again, architectural shortfalls are often discovered late in the development life cycle.

The Architecture-Centric Development Method (ACDM) differs from these extremes in that it places the software architecture, not software processes or code artifacts, at the center of a development effort. Like architectures in the building and construction industries, the ACDM prescribes using the architecture design to drive not only the technical aspects of the project but also the process and programmatic issues of a development effort. The ACDM weaves the product, technology, process, and people together into a cohesive, lightweight, scaleable development method. This presentation gives an overview of the ACDM, briefly discusses experiences thus far in using the method, and maps out plans for maturing the method.

Notes

First, Anthony Lattanze underscored the need for architecture in all software development efforts, not only in big companies but also in medium-sized and small ones. He detailed his method, the ACDM, and emphasized how it helps focus development effort on the architecture. The ACDM is in the middle of two extremes: (1) high-speed development methodologies that ignore architecture and (2) heavy, process-oriented paradigms. Lattanze showed that when used properly, the ACDM helps achieve not only the desired functionality but also the desired quality attributes.

Lattanze created the ACDM in 1999 as a graduate course project and later improved it using the ATAM and QAW. He discussed his experience with companies that adopted this method with good results. His experience showed that, in most cases, it took three iterations to produce sound architecture. Industry wants something that teaches engineers how to architect. Lattanze argued that his method puts architecture at the center of the development effort and makes it the guide for design and implementation. Another motivation behind his method is allowing small companies to reap the benefits of architecture-driven development, something they might think is impossible.

In the Q&A session, the main topic discussed was when to stop architecting. Some audience members liked the observation that it takes about three tries before you get an architecture right. Others debated about when the architecting has been sufficient enough for you to move on to the next phase of software development. Lattanze argued that knowing when to stop depends on the architectural team's understanding and experience.

3.3.3 Architectural Design of an Automatic Guided Vehicle (AGV) Transportation System with a Multi-Agent System Approach

Danny Weyns, Researcher, DistriNet Labs, Katholieke Universiteit Leuven

Abstract

Introduction: Egemin N.V. is a Belgian manufacturer of Automatic Guided Vehicles (AGVs) and control software for automating logistics services in warehouses and manufactories using AGVs. In a joint research and development project, Egemin and the DistriNet research group developed an innovative version of the AGV control system to cope with new and future system requirements such as flexibility and openness. In this project, a multi-agent system approach was applied for modeling and implementing a decentralized control system. Instead of a centralized approach, where one computer system is in charge of numerous complex and time-consuming tasks such as routing, collision avoidance, or deadlock avoidance, in this project, the AGVs are somewhat autonomous. This autonomy enables a system to be far more flexible than in current software. The AGVs can adapt themselves to their immediate environment, and order assignment is dynamic. The system can cope with AGVs leaving the system (e.g., for maintenance) or with adding new AGVs automatically. To develop the AGV application, we used the evolutionary delivering life-cycle model, which centers architectural design within development activities.

Requirements: To describe the functionality of the software system, we worked with the main stakeholders of the system to define scenarios. Some are initiated by an external actor (e.g., a scenario that describes the life cycle of a task that enters the system); other scenarios describe interactions among parts in the system (e.g., a scenario that describes AGV collision avoidance on crossroads). To establish quality requirements, we used quality attribute scenarios. In particular, to elicit quality attribute scenarios, we organized a QAW with the main stakeholders involved in the project. During this two-day workshop, we generated a utility tree to define and prioritize the relevant quality requirements precisely. In particular, we specified quality attributes such as flexibility and openness, which were important project quality goals.

Architectural Design: For architectural design, we used techniques from the ADD method—a recursive decomposition method that is based on understanding how to achieve quality goals through proven architectural approaches. At each stage of the decomposition, we selected architectural drivers together with the architectural approaches needed to satisfy them. We used a reference architecture for situated multi-agent systems extensively as an asset base for selecting architectural solutions. This reference architecture, developed at DistriNet Labs, represents our expertise with the architectural design of various situated multi-agent system applications. The software architecture of the AGV application is documented by different views. Each view belongs to one of the following standard viewtypes: Module, Component-and-Connector, or Deployment.

Evaluation: We used the ATAM for the evaluation of the software architecture. The method's main goal was to determine whether the software architecture satisfied system requirements, in particular the quality requirements. We applied the ATAM for one concrete application—a tobacco warehouse transportation system used as a test case in the project. The one-day ATAM evaluation was a valuable experience. The full group of stakeholders discussed the architecture in-depth for the first time. Participants agreed they gained insight in four areas:

1. the value of software architecture in software engineering
2. the importance of business drivers for architectural design
3. the value of explicitly listing and prioritizing quality attributes with the stakeholders
4. the strengths and weaknesses of the architecture and architectural approaches

One interesting discussion arose from the tradeoff between flexibility and performance. Various decisions in the software architecture aim to improve flexibility in the system, yet the decentralized nature of the multi-agent system implies an increase in bandwidth. Field tests conducted after the ATAM evaluation proved that the communication cost remains under control even in the worst-case scenarios.

During the evaluation, stakeholders made the following comments about the ATAM:

1. A thorough and complete architectural evaluation based on the ATAM of a realistic industrial application is not manageable in one day.
2. Coming up with a quality attribute tree proved to be difficult, time-consuming and—at times—tedious. A lack of experience and clear guidelines on how to construct such a tree hindered group discussion.
3. The general AGV software architecture was developed with several automation projects in mind; however, during the ATAM evaluation, the scope of the architecture was assumed to be a single automation project. Clearly, the ATAM is devised to evaluate a single architecture in a single project. However, this difference in scope hindered discussions because some architectural decisions were motivated by the architecture's product line nature.
4. We lacked good tool support for documenting architectures. Currently, drawing architectural diagrams and writing the architectural documentation incurs much overhead. Revising the documentation and keeping everything current (e.g., cross-references and relations between different parts of the documentation) turned out to be especially hard and time-consuming. In the future, good tool support would be helpful.

Conclusion: Developing the AGV application with a multi-agent system approach was a valuable experience for both partners in this project. Egemin learned a lot about the potential and possible implications of applying multi-agent system technology in AGV systems. At DistriNet, this real-world application showed that

multi-agent systems can make a difference when qualities such as flexibility and openness are important system goals. Finally, we gained insight into the relationship between multi-agent systems and software architecture.

For a detailed description of the AGV application's software architecture and an extensive report on the ATAM evaluation, go to http://www.cs.kuleuven.be/publicaties/rapporten /cw/CW431.abs.html.

Notes

Danny Weyns described how his research group, DistriNet Labs, and Egemin N.V., a Belgium manufacturer, joined their efforts and built an innovative AGV. This research and development effort used many architectural techniques to cope with the new quality attributes of AGVs. Weyns described the functionality of AGVs and their quality attributes (both old and new), emphasizing the importance of two quality attributes: openness and flexibility. The companies used software architecture as a means to achieve these quality attributes and used ADD techniques during the design stage. They also developed a reference architecture for AGV systems. In addition, the group built new middleware to support the development of AGVs. The documentation of the architecture was created using architectural views. Weyns showed some of them and explained some of the architectural decisions made during design.

In the second part of his presentation, Weyns talked about the lessons learned during his experience using software architecture techniques to build AGVs. He argued that documentation consisting only of views was inadequate and that cross-view relationships were needed to communicate the architecture to the stakeholders. He commented that ADD is helpful as a design approach for refinement and would like to see some guidance on using the approach for evolving systems. He also described the ATAM process that was performed to evaluate the reference architecture for these systems. Their experience showed that the utility tree was an important tool but it was time-consuming and required good preparation. A complete evaluation of a complex system such as the AGV system was not manageable in one day. Their attempt to evaluate both the product-line-like basic architecture and a product instance during the same exercise hindered discussion.

3.4 REQUIREMENTS

3.4.1 A Comparison of Requirements Specification Methods from a Software Architecture Perspective

Ipek Ozkaya, Member of the Technical Staff, Software Engineering Institute

Abstract

Not all requirements are equal from the viewpoint of architecting a system. Often, architecturally significant requirements are not specified in a manner that makes them useful to an architect. Exact categorizations of architecturally significant requirements do not yet exist, but given the consensus that those requirements in-

clude quality attribute requirements, we examined how various methods support their expression. We addressed one element of the omission—quality attribute requirements—and evaluated the following methods: natural language requirements using "shall" and "will," use case analysis, the QAW, global analysis, and an approach developed by Fergus O'Brien that we call *O'Brien's approach*. We chose these five approaches because they are in widespread use and/or represent methods that emphasize the capture of quality attribute requirements in particular.

Our ultimate goal is to give guidance as to how to transform the type of business analysis that is done at higher management levels into architecturally significant requirements. Each method's realization differs dramatically from its potential with its own strengths and weaknesses. Only O'Brien's approach explicitly starts with business goals for extracting architecturally significant requirements; however, it focuses on the process without guidance for the specification. The natural language approach (using shall and will) is expressive but does not work well in practice; it often results in a disparate set of requirements that correspond to a collection of "point" requirements. Use case analysis is widely adopted but does not provide enough guidance for quality attribute elicitation and specification. Lastly, there is not enough information about the effectiveness of global analysis and the QAW in practice.

We used the following nine criteria to evaluate the methods:

1. quality attribute expressiveness
2. ease of organizing quality attribute requirements
3. traceability
4. checking for completeness and consistency
5. support for testing
6. tools
7. support for variability
8. skill level needed to carry out the method
9. support for prioritizing requirements

In our analysis, we observed that all the methods

- offer limited assistance for checking the completeness and consistency of quality attribute requirements
- require highly skilled personnel to apply them

We can imagine an ideal method for deriving and expressing quality attribute requirements if we combine the above observations with the best features of each method (identified in parentheses):

- Quality attribute requirements are derivable in a systematic fashion from business goals. (O'Brien's approach and the QAW)

- Quality attribute requirements are expressed in a clear and testable fashion. (QAW scenarios)
- Architecturally significant requirements can be clearly identified. (Use case modeling with a focus on model-driven requirements engineering)
- Requirements derived from organizational factors can also be systematically derived and tested. (Global analysis)
- Educational and explanatory materials would be generally available.

This work was done in collaboration with Len Bass, John Bergey, Paul Clements and Paulo Merson (all from the SEI), and Raghvinder Sangwan (Penn State Great Valley).

Notes

Ipek Ozkaya compared several requirements specification methods to show how well they produce requirements an architect can use to make sound architectural decisions. This comparison was based on the belief that the requirements specification has a great effect on the architecture phase. Ozkaya emphasized the goal of this evaluation: to provide guidance for changing business goals into architecturally significant requirements that will help architects do their job. Ozkaya moved the discussion to the nine evaluation criteria listed in her abstract. One participant asked why investment projection was not used as an evaluation criterion. Ozkaya explained that this factor was part of the process and therefore was not an artifact.

Ozkaya described each method, evaluated it against the nine criteria, and pointed out the strengths and weaknesses of each. Next steps will focus on collecting business stories.

3.4.2 The Best of Three Worlds: Combining the QAW, Model-Driven Requirements Engineering (MDRE), and Global Analysis (GA)

Robert W. Schwanke, Senior Member of the Technical Staff, Siemens Corporate Research

Abstract

The Good Enough Architectural Requirements (GEAR) process is an iterative, incremental analysis process that integrates three approaches to architectural requirements engineering: (1) quality attribute scenarios (as used in the QAW), (2) model-driven requirements engineering (MDRE), and (3) global analysis (GA). GEAR shows where these methods overlap and where they complement each other. It also adds insight into the differences between product requirements and architecture requirements and incorporates experience from over a dozen diverse industrial software architecture projects.

Notes

In his presentation, Robert Schwanke compared three methods of gathering architectural requirements: quality attribute scenarios (used in the QAW), MDRE, and

GA. His analysis led him to create a new approach, GEAR, that combines the best of the three methods. Schwanke described GEAR as an incremental, adaptable, pragmatic, and efficient approach.

Schwanke described the nontechnical issues that face architects at the initial stages of architecture, including competing stakeholders' interests, project management styles, and the uncertainty of project direction and funding. Then, he discussed the "artifact-uses" relation which is analogous to the Parnas "module-uses" relation.

Schwanke provided an overview of the GEAR process. He explained that MDRE helps elicit product requirements in the form of use cases, which focus primarily on functionality. To supplement these requirements, a quality attribute approach contributes quality attribute scenarios. Finally, GA provides an architecture problem analysis that links requirements to architectural strategies. He noted that iteration was not shown in his model for the sake of clarity, but it was implied. Next, he presented an instance scenario of how you could use the process during the first iteration of requirements elicitation and explained that

- stakeholder analysis produces stakeholder scenarios (both use cases and quality attribute scenarios)
- GA analyzes the requirements and then produces architectural strategies
- architecture principles employ the strategies used to produce the decomposition and interfaces of the product architecture

Finally, the discussion moved to the comparison of *requirements* and *factors.* Requirements are true, related to products, unambiguous, verifiable, modifiable, consistent, complete, and traceable. Factors are true, related to product architecture, explicitly variable, arguable, readable, conflicting, important, and eventually traceable. We often generalize factors from requirements. Factors let us conceptualize candidate requirements and are subject to change (unlike the resulting requirements).

3.5 ATAM

3.5.1 Risk Themes Discovered Through Architectural Evaluations

Len Bass, Senior Member of the Technical Staff, Software Engineering Institute

Abstract

The ATAM is a technique for evaluating software architectures to find risks that are linked to the business goals and have architectural implications. The SEI has been doing evaluations based on the ATAM since 1998 and distilling these risks into risk themes since 2000. Risk themes are a summarization and consolidation of the collection of risks found during the evaluation. They are continuously emerging risks that appear repeatedly in the total collection of risks, sensitivities, and tradeoffs, and they have a direct impact on the business drivers and the software archi-

tecture. Figure 1 shows the conceptual flow of the ATAM. Most evaluations produce an Architecture Evaluation Report, which repeats the business drivers used as input to the ATAM and enumerates the risk themes. For this presentation, we use the raw data from those reports of 18 ATAM evaluations conducted by the SEI between 2000 and 2005: 12 are for systems being produced for the DoD; 2 are for systems being produced for non-DoD government agencies; and 4 are for systems being built for commercial purposes. The domains involved range from information systems to embedded systems.

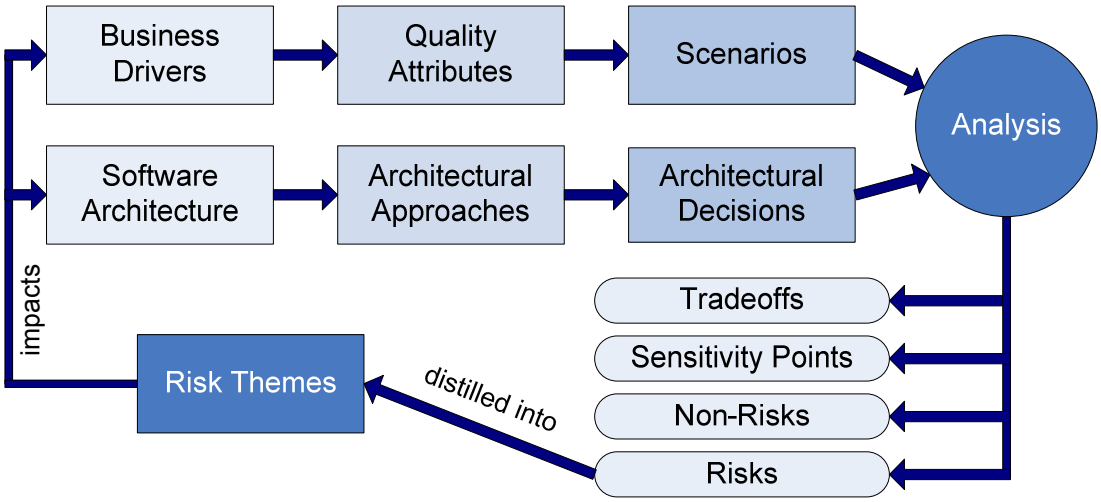

Figure 1: The Conceptual Flow of the ATAM

The major results of this investigation are

- a categorization of risk themes
- the observation that twice as many risk themes are risks of "omission" as are risks of "commission"
- a failure to find a relationship between the business goals of a system being evaluated and the risk themes associated with the development of that system
- a failure to find a relationship between the domain of a system being evaluated and the risk themes associated with the development of that system

We draw lessons for the practitioner and the researcher from the data analysis.

This work was done in collaboration with Robert Nord, William Wood, and David Zubrow of the SEI.

Notes

Len Bass's presentation discussed a preliminary analysis of the results collected from ATAM evaluations conducted by the SEI between 2000 and 2005. He analyzed reports from 18 evaluations and organized the results into risk themes.

To start, Bass discussed the content of the raw data he analyzed. It consisted mainly of risks grouped into risk themes using the Affinity Diagram process. Bass provided examples of each risk theme category. Three of them had subthemes of architecture, process, and organization. Bass also described how he and his colleagues categorized the business goals and then asked questions about the relationships between the risk themes and the business goal categories. Bass stated the two main results of his analysis:

1. Risks of omission prevail over risks of commission.
2. No evidence exists of a relationship between either the business goals and mission goals or the domain and the risk themes discovered during an ATAM evaluation.

Bass presented two of the recommendations he and his colleagues have for practitioners and researchers:

1. Practitioners should use checklists early in the project and use known techniques for mitigating requirement volatility risks.
2. Researchers can explore the hypothesis that risks are related to organizational settings and determine techniques for mitigating the risks of organizational awareness and failure to address important considerations.

Bass also suggested that future work is needed to incorporate business goals and risk themes into ATAM phases. Finally, he asked audience members to help the SEI improve the ATAM by sharing their ATAM evaluation results and observations.

3.5.2 The ATAM and Collaboration at the Enterprise Level

Craig R. Martin, Director, Knotion & Osilio Companies (Pty) Ltd.

Abstract

We discovered a distinct gap between enterprise-level and software-architecture-level planning in our consulting and software development divisions. As a result, we wanted to build a system that would

- dynamically measure what impact a specific functional requirement (realized in a use case) has on a business value driver
- measure the traceability from the “deliver products to market faster” business objective down to a performance metric on an Enterprise Resource Planning (ERP) solution across a low-bandwidth network

To satisfy these objectives, we built the Synap-c knowledge automation suite—a workspace where you can mind-map classified data into a database and associate that data with more structured software engineering information. For example, you could use Synap-c to develop a use case model and link it to goals, objectives, and strategic outcomes. Synap-c allows the usual UML modeling with the ability to associate these models with an enterprise architecture or a simple mind-map ses-

sion held in a boardroom meeting. We also modeled the ATAM into the solution and linked it with both the software-architecture, UML aspect of the tool and the enterprise-architecture-planning components. As a result, we created a bridge between the two domains according to their structured models.

Our main objective was not to capture this information but to share, reuse, and automate it. We can understand how all the components work together to form a risk theme or mitigation tactic. Together, all these components build what we refer to as the *knowledge* of a specific theme or tactic. Synap-c can reuse this knowledge because it can be uploaded to a host server and shared with other architects both inside and outside the company. We also developed an overarching agent framework that can synthesize the resulting components, themes, and tactics into a belief system. Thus, we could have a reliability agent that knows and understands all the rules and components it interacts with and can adjust a solution's reliability accordingly. We can also place these reusable agents into a software development environment where similar belief systems are captured in their own agents (e.g., project schedules, resources, objectives, metrics, and technical hardware). Now we can determine the overall impact of changes to "data primary keys" in a nondeterministic environment, and the reliability agent can learn from its changes in the project environment.

At present, we want to accumulate as much ATAM-type information as possible to build suitable belief systems and to create reusable intelligent and autonomous agents. These agents could support quality attributes at both the software and enterprise levels. The ATAM data could be collected at individual companies or through an open-hosted environment.

Here are some advantages of using Synap-c within the architecture space as well as for evaluations based on the ATAM:

1. Models such as the ATAM can be extended to meet individual requirements.
2. A network structure can be used to map anything to anything, which allows trawling through the entire network.
3. Scenario and gap analysis can be performed on various options.
4. The semantic classification of data and reporting can be performed.
5. Reuse, if needed, is possible at the team, department, client, industry, and international levels.
6. The blackboard nature of the tool is conducive to architecture in practice. There is support for the agnostic nature of architects and developers.
7. A gap still exists between the outputs of architecture decisions and the risks identified in tangible code-level decisions. This gap can be significantly reduced with the use of Synap-c's captured belief systems.
8. Feedback from architecture studies and the post-analysis of ATAM data must also cater to the semantic nature of architecture perception, which is partially

captured in styles. The automation of these styles into belief systems may yield some interesting results.

Notes

Craig Martin's presentation discussed the use of a tool set, Synap-c. His company developed Synap-c to enable collaboration among architects, project managers, developers, and stakeholders at the enterprise level. This collaboration is targeted at project evaluation and planning. To start, Martin discussed the foundation of this tool. Synap-c builds upon the concepts of the ATAM by forming a closed-loop collaboration that keeps all stakeholders informed of the architectural decisions and relays their feedback.

Martin discussed in detail how he and his colleagues used the ATAM within Synap-c and used both logical and physical views to illustrate the process. He then presented some of the results of using the ATAM with Synap-c (which are reported in the abstract). He said that future work would focus on how to automate the rules (concepts) of the ATAM within this tool.

4 Working Sessions

The working sessions are meant to engage leading-edge software developers, acquirers, and researchers in identifying emerging solutions to pervasive problems. Six working sessions were scheduled to provide further discussion of topics related to software architecture:

1. Architectural Competency
2. Bridging System and Software Architecture
3. Architecture Evolution
4. Global Software Development
5. Strategic Risk Management for Architectures
6. Building a Software Architecture Community

Participants were asked to describe the topic, discuss why it is important, define the gaps between what technology offers and industry needs, and discuss possible solutions.

These summaries are meant to convey preliminary ideas for the purpose of getting feedback. They do not necessarily represent the consensus of the session participants.

4.1 ARCHITECTURAL COMPETENCY

Moderator: Len Bass, Software Engineering Institute

Architectural competency is a term that conjures up different visions. One goal of this breakout was to determine what meanings the attendees envisioned. One aspect of architectural competency is the competency of an individual versus that of the organization. Another aspect is an architect's required technical skills versus nontechnical skills.

We began by distinguishing between the organizational perspective and the individual perspective. For an organization to be architecturally competent, it must have a number of competent architects, but this staff alone is insufficient. One indicator of architectural competence for an organization—in addition to the presence of competent architects—is the organization's established set of practices. For example, regular architectural reviews should occur. Program plans and schedules should reflect the influence of architecture (e.g., work teams are not assigned until at least the first-level decomposition of the architecture has been defined). There should be a career path for architects and an architectural governance system to establish corporate standards and practices as they refer to architecture. This governance system might establish, for example, the views and notations by which an architecture should be documented.

The interests of the attendees at the breakout group cut across the individual and the organizational perspectives. The interests were

- How should software architects be certified?
- How should software architecture be mapped to early life-cycle activities?
- How can an organization develop a roadmap to improve its architectural competence?
- What goes on in an architecturally competent organization?
- What is the skill set of an architect?
- How should an organization judge talent to determine who should become architects?
- How do organizations gauge architectural competency?
- What do architects do on a daily or project basis?
- How do you become a competent architect?
- How can an architect mentor others to become competent?
- How does an architect influence an organization to become more architecturally competent?
- How can the architectural competency of a team be assessed?

Attendees of this session were practitioners and researchers from the following organizations: Chemical Abstract Service, CIBER, the SEI, Siemens, Union Switch and Signal, and Visteon.

The discussion primarily centered on two areas: (1) the duties of an architect and (2) the skills of an architect.

4.1.1 Duties of an Architect

An architect's duties as articulated by the participants in the breakout group include

- problem solving
- project planning
- establishing an architecture and a design for a project
- developing processes and procedures for a project or organization
- writing proposals
- defining platforms for the future
- translating stakeholders' needs into requirements
- helping business people understand each other's needs and devising technical solutions to reflect this understanding
- enforcing the use of the architecture within the development team

4.1.2 Skills of an Architect

Architects must have the following attributes:

- organization skills to balance multiple tasks
- communication skills to convey the business goals to various stakeholders
- political skills (including negotiation skills) to convince various stakeholders of the wisdom of a technical choice and to navigate within an organization's political factions
- a broad view of the
  - product
  - organization
  - mission and goals that the product is intended to support
- experience to draw upon when making decisions
- leadership abilities to mentor and inspire others
- technical depth to understand and solve complicated technical issues
- the right perspective to plan for the long term and to identify what is and is not important to the architecture

4.1.3 Next Step

The discussion focused on an architect's duties and requisite skills. These items were not discussed:

- how to map skills to roles and duties
- what an organization must do to become architecturally competent

Given the attendees' interest in organizational competence, it might be the topic of papers or a breakout group at the next SATURN workshop.

4.2 BRIDGING SYSTEM AND SOFTWARE ARCHITECTURE

Moderators: Mike Gagliardi and David Zubrow, Software Engineering Institute

There is currently a gap between the engineering practices of system architecture and software architecture. This disconnect causes many problems in the development and acquisition of large-scale, software-intensive systems in the DoD. Systems-of-systems (SoSs) are particularly susceptible to major disconnects between system and software architectures. An SoS depends on two things: (1) the system architecture to guide the development of individual systems and the concept of operations (CONOPS) and (2) the software architecture allowing interoperation between nodes and the sharing of critical timely information. All programs must find a way to match the system architecture to the software architecture, to handle the inevitable development parallelism, and to ensure that the system architects, software architects, and CONOPS developers interact to get the best possible system. This workshop was a facilitated forum for capturing the current state of practice for

integrating system and software architectures. We also identified the architecture integration gaps and technical obstacles.

Discussion questions included

1. How do you manage the system's quality attributes within and between the system and software architecture(s)?
2. How do you describe the mapping between the representations of the operational architecture, system architecture, and software architecture representations? How do you relate the views in the architectures?
3. What are the risks of and lessons learned in the integration of system and software architecture practices?

Attendees of this session were practitioners and researchers from the following organizations: Boeing, Navy, Northrop-Grumman, Pitney-Bowes, Raytheon, Rolls-Royce, the U.S. Army, and Virginia Tech.

4.2.1 Why Is This Disconnect Important?

Software-intensive systems often suffer severe integration and operational/behavioral problems due to a lack of consistency between the system and software architectures in addressing system quality attributes. This deficit often results in costly rearchitecting/redesign efforts and operational failures that significantly impact the system's cost, schedule, and mission effectiveness.

4.2.2 Current State of the Practice

Technical

- quality attribute requirements and specifications
  - Quality attribute requirements are often underspecified.
  - It is difficult to identify requirements and quality attribute omissions at the system level.
- system architecture representations, analysis techniques, tactics, and so forth
  - Component relationships at the system architecture level don't have the robustness that occurs at the software level.
  - The tactics for many quality attributes are not described at the system level.
  - There is no analytical support for quality attributes in many instances.
  - No real techniques are available to reason beyond the well-known quality attributes.
  - Systems architecture equals system hardware block diagrams. There is little notion of multiple system views (hardware maintainability, sustainability, etc.).
  - The DoD Architecture Framework (DoDAF) doesn't incorporate software concerns. The levels of abstraction hinder communication between the system and software architectures.

- metrics
  - The lack of metrics about other important quality attributes makes reasoning about them difficult.
- common semantics
  - There is no common language for the information models in system and software groups. Software developers should attend system design meetings to enable cross-pollination.
  - System architects understand quality attributes but in a different terminology.
- architecturally significant scenarios
  - Software is becoming scenario based, which creates a bigger disconnect with some system engineers.
  - Mission threads are a good starting point for system scenarios (i.e., sustainment, availability, and performance).
  - Scenario templates are used in activities for system fault-tree analysis.

Process

- Decisions at the system architecture level are often pushed down to lower software architecture levels.
- There is a lack of communication between system and software groups.
- System architecture is the result of a waterfall process with functional decomposition.
- System architects don't fully understand and address all of the quality attributes.
- Early and frequent integration and testing saves integration effort in common practice, but these tasks may not scale in a large, distributed SoS context.
- Typically, system architecture is defined before hardware and software trade-offs are made.

Programmatic

- Time and cost constraints make it difficult to get to the quality attributes.
- Customers drive the decision-making process. Companies need to follow architecture-centric acquisition strategies.
- A lack of return on investment (ROI) data impacts management's level of commitment to architecture activities.
- Developers and acquirers are pressured to allow prototypes to become products.
- The system engineering (and acquisition) community lacks an awareness of the QAW and ATAM.
- System engineers/architects need to be trained and added to ATAM evaluation teams.

Organizational and Cultural

- A separation of systems and software groups often occurs.
- Software groups are large (5×) compared to system groups.
- There is no real system-architecting team.
- There is a lack of respect between the systems and software groups. Parallel learning experiences create different cultures.
- Postmortem reviews are the current means for improvement, but the findings are rarely applied.

4.2.3 Technical Gaps and Issues

Gaps

- quality attribute requirements and specifications
- system architecture representations, analysis techniques, tactics, and so forth
- quality attribute metrics
- common semantics
- system scenarios

Issues

- The strict, centralized SoS architect/architecture concept is in question in this context.
- Can the ATAM be moved into the system area? Some concerns that need to be addressed include scale, decomposition, scenario/mission threads, and schedule.
- How service-oriented architecture (SOA) and Web services impact architecture (and vice versa) but are not well understood.
- How open source approaches impact architecture (and vice versa) and must be better understood.
- Intersystem policies must be thought of as the next higher level of abstraction; however, technology may be moving too fast to fully cover those policies.

4.2.4 Next Steps

The moderators developed the following recommendations for next steps after the working session. These recommendations do not necessarily reflect the views of the session participants.

Near-Term Steps

- Collaborate with external organizations to understand their needs and any extensions they've made to the QAW, ATAM, and so on for system architecture.
  - Identify any necessary extensions they've made to existing SAT methodologies.
  - Initiate pilots and case studies with external collaborators.
- Interview individual workshop participants.

- Have system engineers/architects attend classes in the SEI's Software Architecture curriculum.

Long-Term Steps

- Actively participate in the following groups and standardization discussions (among others) to ensure that they address quality attribute concerns adequately:
  - The International Council on Systems Engineering (INCOSE) to gain more insight and influence in its architecture framework activities
  - SysML standardization and quality attribute UML extensions
  - The DoDAF
  - The Object Management Group (OMG)
  - The Open Group Architecture Framework (TOGAF)
- Describe tactics, patterns, and other features at the system level for the remaining important quality attributes. Collaboration with external organizations is necessary.
- Use system scenarios with mission threads as a good starting point. We need to
  - augment the scenarios for all the important quality attributes
  - transform the scenarios into software-specific scenarios
  - collaborate with external organizations and participate in pilots
- Investigate how SOA contributes to and impacts operational-to-software transformations.
- Collaborate with external organizations and staff in the SEI's Software Engineering Measurement and Analysis (SEMA) Initiative regarding quality attribute metrics.
- Leverage the academic community to get methodologies such as the QAW and ATAM into the systems engineering process.
- Determine how to address common semantics (a technical and cultural issue).
- Investigate the role, activities, and other aspects of SoS architecture based on a more decentralized approach at the SoS level.
- Hold another working session at the next SATURN workshop.

4.3 ARCHITECTURE EVOLUTION

Moderators: Felix Bachmann, David White, Software Engineering Institute

The vision of the SEI Architecture Evolution project is to provide an effective, integrated, widely applicable, and extensible set of life-cycle architectural practices and tools. An architect can use these practices and tools to keep software architecture in line with its goals as the system evolves.

The challenges associated with this vision include

- understanding the current state of the architectural design. This task entails discovering the discrepancy between the current and desired states of the architecture.
- finding the appropriate practice and/or tools that can take the architecture where it needs to go. This task may involve tailoring and integrating practices, and understanding the appropriate fit with other architectural processes and technologies.

During this two-hour working session, we sought to

- elicit from the participants situations in which they have evolved the architecture
- identify existing practices and tools that are used to solve problems
- identify gaps in those practices and tools
- review opportunities for further work

We asked participants to consider three questions:

1. Where are you?
   - What is the current state of the architecture and the forces that impact the architecture?
   - Explore evaluation, reconstruction, and documentation techniques to understand the architecture.
2. Where do you want to go?
   - What new business opportunities do you want to exploit and what risks need to be mitigated?
   - Explore real options and utility techniques to understand alternatives and characterize the benefits of various qualities.

3. How do you get there?
   - How do we change the system, and what architectural strategies provide the most benefit given the cost?
   - Explore design methods to transform the architecture and cost benefit analysis to choose alternatives.

Attendees of this session were practitioners and researchers from the following organizations: ABB, Chemical Abstracts Service, Cherokee Information Services, Katholieke Universiteit Leuven, Knotion Consulting, Mellon, Northrop-Grumman, the SEI, Siemens, and UPS.

4.3.1 Discussion

It seems that architecture evolution is not a problem if it is anticipated and the architecture is prepared for it. In that case, evolution becomes a "simple" change. Unanticipated evolution is a hard problem, since it involves having to evolve software architecture to accommodate unanticipated changes.

Participants responded either that they discovered the need for evolution during architecture evaluation based on the ATAM (risks themes) or that during integration (mismatch of components) or the change was obvious (e.g., new business strategy, new technology, new feature, deploying in the field).

Participants offered the following reasons for failing to anticipate changes: short-term thinking (let's work only on the next version), lack of expertise (they could not predict likely changes), and lack of time and/or money.

Is it possible to avoid evolution? If we could reliably predict everything that could happen and prepare the architecture to accommodate the changes, would that make the evolution problem disappear? In the real world, it isn't feasible to prepare the architecture for every change due to time and cost limitations. Furthermore, there is the risk of overengineering (the introduction of too much overhead or "analysis paralysis") and the inevitability of things happening that no one thought could.

Given that evolution is unavoidable, what can be done when evolution occurs? One of the first activities should be to determine the reasons why a change was not anticipated or why a mismatch occurred. This lack of forethought could be due to a lack of expertise, technology, and/or time.

Next, the scope of the evolution must be determined, and the process of change must be aligned with that scope. Problems outside the scope should not be fixed. The scope can be understood at the micro or macro level (e.g., one product or multiple products involved). The scope can be understood through the use of scenarios to reason about a change. Since changing the architecture is costly, ad hoc decisions can (and should) be avoided through a rigorous process for making decisions involving a Change Control Board (CCB) or ARB.

The participants offered the following guidance on how to decide what to do:

- Architectural redesign: Follow a strategy that defines how to fix the architecture. Make sure everything else is still as expected (conduct architecture regression testing using the scenarios).
- The context in which the architecture lives: Increase your organization's expertise through training and new hires. Keep in mind that it can be a challenge to change people's behavior. Consider the infrastructure required for evolution and whether the organization can adapt to an architectural change. If the organization cannot accommodate certain types of changes, do not make them.
- The time frame: There was a strong lobby for incremental change; don't try to do everything in one step.

4.3.2 Next Steps

In summary, the major points for the session were

- Do as much as appropriate and feasible to avoid unanticipated change.
- Use architecture evaluations, such as ATAM evaluations, to discover discrepancies.
- Use quality attribute scenarios to
  - guide the evolution
  - define the scope for the evolution
  - guide the change process of the architecture
  - ensure that all parts of the architecture not involved in the evolution still have their required properties

The SEI is interested in collecting information from the developer community about additional situations in which they have evolved the architecture, the problems they encountered, and best practices for solving those problems.

4.4 GLOBAL SOFTWARE DEVELOPMENT

Moderators: Matthew Bass, Siemens; David Zubrow, Software Engineering Institute

The trend towards global software development has quickened pace in recent years. More and more software-intensive systems are being developed using teams that are geographically distributed. Developing software in this way poses unique challenges. Not only do cultural issues, background differences, and organizational boundaries come into play, but day-to-day communication and coordination is much more difficult and less effective. The system architecture is a central artifact in these efforts. The goal of this working session was to bring together people who are concerned with architecting systems for global development. During this working session, we collected our experiences to identify issues that are unique to global software development.

Participants were asked to consider the following questions:

1. What software architecture issues have you experienced while developing software in teams that were geographically distributed?
2. What practices were effective in this environment?
3. What practices were ineffective in this environment?

Attendees of this session were practitioners and researchers from the following organizations: Bosch, SEI, Siemens, the U.S. Army, Virginia Polytechnic Institute and State University (Virginia Tech), and Visteon.

The participants expected to learn about the following:

- other attendees' experiences with Global Software Development (GSD)
- assurance of GSD
- lessons and strategies for GSD
- outsourcing
- communication techniques for GSD
- tools for project management and collaboration for GSD
- architecture for GSD
- cost implications of GSD
- metrics for GSD
- effective collaboration across sites

4.4.1 Discussion

The participants from Visteon talked about their experiences and issues with GSD projects. Others in the room shared their experiences and identified some strategies for dealing with those issues. These discussions are summarized in Table 2.

Table 2: List of GSD Issues and Strategies

Issues	Strategies
Poor quality of code; team in India doesn't have good technical background.	Coordinate joint reviews across teams. Staff in India must work in the U.S. during the project's final stages. Afterwards, they should return to India and become more independent.
Short-term use of resources for managing remote teams	Implement frequent builds with automated testing.
Inexperienced remote team members; high turnover; they don't develop domain knowledge.	Give remote teams incentives to stay in the company (e.g., the most productive team/person could come to the U.S. for a while).
No social relationship with remote team	Develop remote sites as competence centers to reduce turnover.

Table 2: List of GSD Issues and Strategies (Continued)

Issues	Strategies
The lead engineer doesn't produce very rich design documents.	Create better documentation.
Some phases (e.g., design) should happen with teams collocated. That represents a much higher cost when the team is in both the U.S. and India.	Have two low-cost centers and let them compete for jobs.
The time zone difference is a major problem; distribution itself is a major problem.	• Constrain the development of remote teams using a tool infrastructure. • Do testing independently from development. • Involve remote teams in early life-cycle activities (e.g., the QAW, ADD, and the ATAM). However, it's not a good idea to have stakeholders participate in an ATAM evaluation via video teleconferencing because part of the evaluation's value comes from the social interaction. • Conduct cross-cultural training. • Move some local team members to the remote site and vice versa. This cross-team fertilization can improve trust and minimize attitude issues. • In some low-cost countries, people would do better if you specify exactly what they are to do, that is, if you lay out each step for them. • Use good tools to support configuration management, integrated builds, and a well-defined process.

The discussion focused on the attendees' problems and their strategies for dealing with them. It was clear there were many interrelated concerns such as

- system/software architecture
- architecture documentation
- organizational practices
- work allocation

4.4.2 Next Steps

How the concerns discussed in the previous section influence each other is not known. In the future, it could be fruitful to gain an understanding of the factors that come into play in GSD projects and their interrelationships. This exploration of GSD factors could be the basis for future research and/or breakout sessions.

4.5 STRATEGIC RISK MANAGEMENT FOR ARCHITECTURES

Moderator: Rick Kazman, Software Engineering Institute

The purpose of risk modeling is to aid in risk management decision making. Managing risk does not necessarily result in removing it; zero risk is not always possible or even economically feasible. For any risk, there is usually a limit to how much it can be controlled or mitigated. As such, *risk management* is the collection of activities used to address the identification, assessment, mitigation, avoidance, control, and continual reduction of risks. The goal of risk management is "enlightened gambling" where we seek an expected outcome that is positive regardless of the circumstances.

In general, architectures are not well planned or managed. Frequently, they are poorly documented and not analyzed. As such, their impacts on the future and on their stakeholders are not considered. Why is that the case? Frequently, the focus of an architecture team is on technical matters—where the value is perceived to be. Nontechnical factors such as risk, opportunity, and cost are considered boring or "management issues." Yet, despite recent progress, techniques for analyzing and assessing risk in architectures are not widely known or taught.

The purpose of this working session was to assess the state of the art in risk management for architectures with an eye towards improvement. The perspective that we took in the workshop is as follows: The expected value of an architecture is the difference between its expected gain and its expected loss:

$$\begin{aligned} E[\text{Value}] &= E[\text{Gain} - \text{Loss}] \\ &= E[\text{Gain}] - E[\text{Loss}] \end{aligned}$$

To increase the expected value, we can only increase the expected gain or reduce the expected loss (subject to cost). Strategic software engineering methods explicitly optimize the expected value with respect to cost. To maximize the expected value, we need to manage both opportunities (expected gain) and risks (expected loss). Risks are situations or possible events that can cause a project to fail to meet its goals; those risks range in impact from trivial to fatal and in likelihood from certain to improbable. Opportunities are the potential to realize benefit from a project. They are analogous to risk but have a beneficial impact. Both risk and opportunity are quantifiable. Risk is typically quantified as "risk exposure" (similarly for opportunity):

Risk Exposure: $RE = Prob(Loss) * Size(Loss)$

Opportunity Potential: $OP = Prob(Gain) * Size(Gain)$

For multiple sources of gain (or loss), this equation applies:

$$RE = \Sigma_{sources}[Prob(Loss) * Size(Loss)]_{source}$$

That is, risk exposure (RE) is the sum, over all sources, of the individual risk exposures. Given our formula for value above, we see that one way to maximize value is to minimize risk (the expected loss).

The basic driving principle for value-based activities is the following: If it's risky to do something, *don't do it* (e.g., specify firm graphical user interface [GUI] requirements too early). Likewise, if it's risky not to do something, *do it* (e.g., specify document-sharing protocols). This rule seems obvious, but it is rarely explicitly followed or managed. People must be educated to perform effective risk identification, assessment, mitigation, prioritization, and tolerance. Part of that education must include learning about models for strategic reasoning about risk. The model discussed in this workshop was one of strategic planning.

Attendees at this session were practitioners and researchers from the following organizations: Bechtel Bettis, Chemical Abstract Service, Cherokee Information Services, Mellon Financial, Northrop-Grumman, the SEI, Siemens, Union Switch and Signal, the University of York, and Visteon.

4.5.1 Strategic Planning

To strategically manage architectures—and in particular to manage architectural risk—we need the following information:

- a set of project attributes that we want to manage (and presumably optimize)
- an estimate of `Size(Loss)` and `Prob(Loss)` for each attribute
- a set of attribute assessment techniques
- a set of assessment cost estimates
- a set of `Prob(Loss)` estimates *after* the application of technique `T`

Given this information, we can determine an optimal management plan for reducing risk and the expected loss, subject to cost. This was the practical problem for the workshop: Which techniques (T1, T2, T3, …) should be used on which attributes (A1, A2, A3, …) and in what order, to reduce an architecture's overall RE? We assume that a technique has a particular cost when applied to a given attribute, that a technique will reduce the RE when applied to a given attribute, and that sometimes a technique may not be used with a given attribute.

Examples of project attributes that are critical for architecture include

- A1: Worst-Case Performance (i.e., priority inversion, queue overflows)
- A2: Availability/Robustness (i.e., no single point of failure)
- A3: Ease of Integration
- A4: Usability
- A5: Performance (i.e., no missed data frames)
- A6: Cost
- A7: Development Schedule

- A8: Portability/Replaceability
- A9: Maintainability
- A10: Scalability
- A11: Testability
- A12: Understandability
- A13: Resource Utilization
- A14: Security

Examples of assessment techniques that might reduce architectural risk include

- T1: SAAM
  (the SEI Software Architecture Analysis Method)
- T2: ARID
  (the SEI Active Reviews for Intermediate Designs)
- T3: FRAP
  (the Facilitated Risk Analysis Process)
- T4: Model Checking
- T5: ATAM
- T6: ALPSM
  (Architecture Level Prediction of Software Maintenance)
- T7: ALMA
  (Architecture-Level Modifiability Analysis)
- T8: OCTAVE®
  (the SEI Operationally Critical Threat, Asset, and Vulnerability Evaluation)
- T9: QAW
- T10: Markov Modeling
- T11: CBAM
  (the SEI Cost Benefit Analysis Method)
- T12: RMA
  (Rate Monotonic Analysis)

We assumed that, for each attribute, we could estimate the probability of a loss due to that attribute and the size of that loss. This information might be captured in terms of dollars, utility, or a percentage of the project value, as shown in Table 3.

® OCTAVE is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

Table 3: Loss Potential and Probability for Architectural Attributes

Attribute *i* (A*i*)	**A1**	**A2**	**A3**	**A4**	**A5**	**A6**	**A7**	**A8**	**A9**	**A10**	**A11**	**A12**	**A13**	**A14**
Loss potential (A*i*)	100	90	90	80	60	30	50	20	10	10	60	10	90	60
P_{before}(A*i*)	6	5	20	15	20	5	20	10	10	10	30	20	50	40

Similarly, we assumed that, for each attribute assessment technique, we could estimate the cost of applying technique T*j* on assessing attribute A*i*, as shown in Table 4 (the letter x means that the technique does not apply to the attribute).

Table 4: Cost of Assessment Techniques Applied to Architectural Attributes

Cost of assessing A*i* with T*j*	**A1**	**A2**	**A3**	**A4**	**A5**	**A6**	**A7**	**A8**	**A9**	**A10**	**A11**	**A12**	**A13**	**A14**
T1	50	x	10	70	10	x	x	x	x	50	5	x	10	x
T2	100	x	x	100	100	x	x	x	x	x	x	x	x	x
T3	x	x	80	80	80	x	x	x	x	x	x	x	x	x
T4	100	90	x	x	19	x	x	x	x	x	x	x	x	x
T5	70	100	70	70	70	x	x	x	x	x	x	x	x	x
T6	30	30	30	30	30	x	x	x	x	x	x	x	x	x
T7	x	x	x	x	x	5	10	x	5	5	3	x	3	x
T8	x	x	x	x	x	80	70	x	80	80	x	x	x	x
T9	x	x	x	x	x	x	3	10	20	20	20	10	20	10
T10	60	x	x	60	50	40	50	50	50	40	40	20	40	20
T11	60	x	90	60	60	x	x	x	x	50	10	x	10	x
T12	x	x	x	x	x	5	5	10	10	10	10	5	x	x
T13	30	x	x	30	30	x	x	30	x	30	5	x	30	x
T14	100	x	x	100	100	x	x	x	x	100	5	x	100	x

Finally, for each technique applied to each attribute, we need to know how much risk is reduced. In other words, we apply a technique to reduce risk in the architecture. To strategically manage risk, we must be able to assess the degree to which a technique mitigates a particular risk. This assessment is captured in Table 5.

Table 5: Probability of Loss for Architecture Attributes After Technique Application

$P_{after}(Ai)$ using T*j*	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14
T1	4	x	15	12	15	x	x	x	x	5	15	x	20	x
T2	6	x	x	13	15	x	x	x	x	x	x	x	x	x
T3	x	x	15	12	13	x	x	x	x	x	x	x	x	x
T4	6	0	x	x	19	x	x	x	x	x	x	x	x	x
T5	6	2	2	13	18	x	x	x	x	x	x	x	x	x
T6	6	2	5	13	19	x	x	x	x	x	x	x	x	x
T7	x	x	x	x	x	2	15	x	8	10	30	x	30	x
T8	x	x	x	x	x	1	10	x	7	9	x	x	x	x
T9	x	x	x	x	x	x	10	4	6	8	25	20	30	30
T10	6	x	x	12	19	3	15	8	8	8	27	20	30	20
T11	3	x	15	5	5	x	x	x	x	5	5	x	5	x
T12	x	x	x	x	x	3	18	9	10	10	30	20	x	x
T13	5	x	x	12	15	x	x	5	x	6	20	x	28	x
T14	3	x	x	3	5	x	x	x	x	5	10	x	20	x

Given this information, a project leader may then choose the type and order of risk-reduction strategies. The choice of strategy matters, as shown in Figure 2.

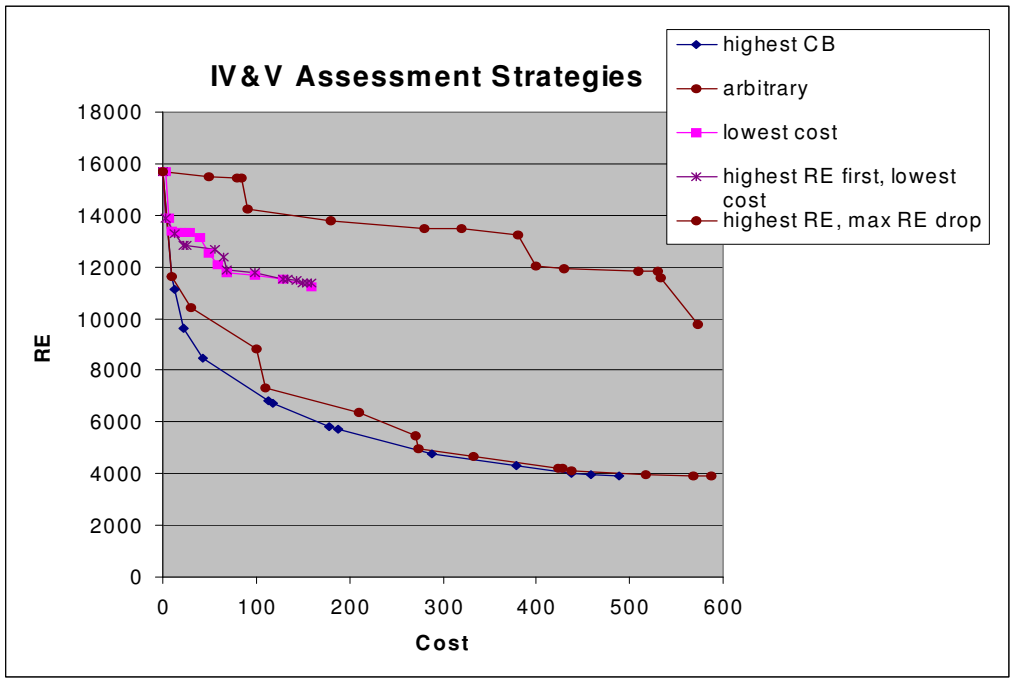

Figure 2: Comparing the Choice of Strategies for IV&V Risk Assessment [1]

Figure 2 shows how different strategies affect risk reduction. The risk-reduction example shown is taken from the domain of Independent Verification and Validation (IV&V). However, the principle is the same for any set of risk-reduction ac-

[1] CB stands for cost/benefit.

tivities: A project decision maker should arrange risk-reduction activities in the order that provides the most benefit per unit cost. To do this for architecture, we need to know the costs and benefits of architecture risk-reduction methods (such as those listed above), as exemplified by the data contained in Table 1–Table 3.

4.5.2 Workshop Output

Eliciting a complete set of data, such as that presented in Table 1-Table 3, was not feasible for the limited time of the workshop. Instead, we focused on eliciting a set of project attributes, a set of techniques that can assess each attribute, and cost information on a subset of the techniques. Establishing more precise models may be a future project. The elicited information on architecture attributes and their risk-reduction techniques is presented in Table 6.

Table 6: Elicited Project Attributes and Their Risk-Reduction Techniques

Attribute	Assessment Technique
Security	• Security checklist in the DoDAF • ATAM-like reviews with scenario generation and analysis only focusing on information assurance • Boeing's Predictive Analysis, Security Method (PASM) (a largely DoD-based checklist style for qualitative security assessment)
Project Management	• Time-box scheduling • Scope reduction • Periodic recomputation of the cost of project completion and how long it will take so that you can address schedule and cost risks and determine how many resources are left
Testability	• Scenario-based testing
Performance	• Boeing's Reused Architectural Component Method (RACM) for changing or new technologies • Boeing's Predictive Analysis, Performance Method (PAPM) for performance and scalability • Instrumentation • Modeling Tools (e.g., SLAM-2) • Building executable architectures with stubbed components to look for risks • Simulation • Experimenting for performance and scalability
Availability	• Boeing's Predictive Analysis, Availability Method (PAAM) for availability analysis • Experimenting for availability
Safety	• HazOp analysis • Fault-tree analysis

Table 6: Elicited Project Attributes and Their Risk-Reduction Techniques (Continued)

Attribute	Assessment Technique
Interoperability	• Inspections for measuring interoperability (which include looking at data exchanges)
Modifiability	• Checklists for modifiability • Experimenting for modifiability
Usability	• Rapid application development • Goals, Operators, Methods, and Selection Rules (GOMS) modeling • Paper prototypes and Visual Basic mock-ups

4.5.3 Next Steps

The workshop, while fruitful, underlined the need for more careful, methodical data collection on risk exposures and risk-reduction techniques and their costs. Some of the workshop participants have agreed to collaborate on defining and collecting the necessary data. That collaboration has already begun.

4.6 BUILDING A SOFTWARE ARCHITECTURE COMMUNITY

Moderator: Rob Wojcik, Software Engineering Institute

During this working session, we discussed our experiences with building a software architecture community of practice within an organization. We also talked about our ideas for growing SATURN beyond a yearly workshop and into a network that would available to members throughout the year.

Prior to the working session, we asked participants to consider the following questions:

1. Who are the stakeholders in your organization who have an interest in software architecture?
2. How does software architecture information flow in your organization?
3. Where do you go for information on software architecture?
4. What opportunities for exchanging ideas would you like to see?

Attendees of this session were practitioners and researchers from the following organizations: ABB, Boeing, Ciber, Katholieke Universiteit Leuven, Knotion Consulting, Mellon Financial, and the SEI.

4.6.1 Types of Software Architecture Communities

We began the working session by discussing what software architecture communities exist today. The specific communities mentioned were diverse, including some that were organizational, industry wide, user related, project related,

large, and small. The discussion eventually shifted from naming specific software architecture communities to listing general types of communities as follows:

- research
- training
- professional (e.g., data management professionals, project management professionals)
- vendor-related (e.g., Java 2 Enterprise Edition [J2EE], .NET)
- organization, project, TIGs
- partnerships
- specialists
- domains (e.g., enterprise architects, workflow)

Participants noted that every community has its own vision, goals, and objectives. Members within a community also have their own goals to improve their own practices which collectively advance the state of the practice.

4.6.2 Community Stakeholders

Having noted some of the general types of software architecture communities, the group focused its attention on listing stakeholders who have a vested interest in communities:

- developers
- project managers
- business units
- software architects
- business sponsors
- enterprise-level architects
- users of the software systems
- researchers
- software vendors (e.g., tools, applications, Microsoft or Sun Microsystems certification)

4.6.3 Community Essentials

Given various types of communities and stakeholders, workshop participants named the following characteristics as essential to successful software architecture communities:

- community vision
- community goals and objectives
- ability to measure progress

- leaders for bringing people together
- organization/structure, responsibilities (e.g., roles)
- people who want to improve their knowledge and skills
- participation and contribution by community members
- sharing information within the community
- funding
- a critical mass of like-minded thinkers with mutual interests

4.6.4 Community Goals and Objectives

Having identified community goals and objectives as key factors in any successful software architecture community, workshop participants identified the following candidate goals and objectives that any community might wish to consider:

- Advance the practice of software architecture.
- Meet with others and share ideas.
- Support users of software architecture.
- Support a community in moving from suppliers to users (e.g., value chain).
- Identify, understand, and fulfill the needs of software architects.
- Identify, understand, and fulfill the needs of stakeholders based on the vision, goals, and objectives.
- Implement knowledge transfer, including education for novices.
- Enforce the consistency of practices (e.g., leading to standards).
- Evolve/mature practices and have a method to assess maturity.
- Establish a network.
- Identify and evaluate new forms of architecting software solutions.
- Share methods and ideas with like-minded people.
- Educate others about our methods and ideas.
- Develop and manage a network of like-minded people.
- Apply new ideas and methods through the network.
- Measure the implementation.
- Gather feedback from applying new ideas and methods.
- Standardize our methods and ideas.

4.6.5 Formulating an Action Plan

Next, workshop participants pondered how to use these discussion results as a basis for formulating an action plan to further the interests of any software architecture community.

An action plan can be derived based on answers to the following questions:

- Who are the key stakeholders in the community?
- What is the vision sought by the community?
- What goals and objectives follow from the community vision?
- What actions follow from community goals and objectives?
- Who will provide leadership within the community?
- What organizational structure is needed to support the community?
- How will information, knowledge, and skills be shared among community participants?
- How will participants be able to contribute information, knowledge, and skills to the community?
- How will community activities be funded?
- How will progress towards the community's vision, goals, and objectives be tracked and measured?

4.6.6 Example: The Software Architect Community

To conclude the working session, participants wanted to apply what they had learned to formulating an action plan for furthering the interests of the software architect community. There was only enough time to answer the first three questions posed in the previous section:

1. Who are the key stakeholders in the community?

   Software architects are the key stakeholders.

2. What is the vision sought by the community?

   Advance the practice of software architecture.

3. What goals and objectives follow from the community vision?
   - Identify and evaluate new forms of architecting solutions.
   - Share the methods and ideas with like-minded people.
   - Educate others about our methods and ideas.
   - Develop and manage a network of like-minded people.
   - Apply new ideas and methods through the network.
   - Measure the implementation.
   - Gather feedback from applying new ideas and methods.
   - Standardize our methods and ideas.

4.6.7 Next Steps

Participants looking to start their own software architecture communities noted that the generated lists of community essentials, goals, and objectives are a good place to start and can serve as checklists. The goals of SATURN can also be critiqued with respect to these criteria to better serve the group's members.

Much of the discussion on achieving the higher level goal of building a community revolved around the issue of knowledge transfer: research, teach, learn, and build networks. What currently works well in knowledge transfer is on-the-job training, involvement in existing projects, and opportunities to practice software architecture and learn from experienced architects. Obstacles to knowledge transfer include cost and issues related to proprietary information and intellectual property. More work is needed in the areas of semantics, ontology, standard terminology, and the effective communication of architectural decisions and the context in which they are made.

5 Closing Session

In the closing session, Linda Northrop, director of the SEI's Product Line Systems Program, led participants in a discussion on emergent themes, workshop highlights, and the future of SATURN.

Participants noted the following key topics:

- the future plans of the SEI's SAT Initiative
- overall integration of software-architecture-centric methods and techniques
- experiences that others shared in applying the methods and taking the next step of transitioning them for use

Participants noted the following highlights:

1. presentations giving evidence of methods in action
2. a comparison of multiple ATAM evaluations and cross-wise analysis
3. the workshop format of the SATURN meeting and the collaboration it fostered
4. a good degree of interaction in presentations
5. a good mix of academic and industry perspectives
6. a sharing of workshop results at the end of the meeting

Participants expressed interest in these areas and requested that the SEI investigate them:

- SOA testing techniques
- integrating methods across the life cycle; for example, moving from the QAW to an ATAM evaluation
- architecting the semantic Web and ontology application
- variability, reference architecture, software product lines, and the ATAM in the context of product lines
- reliability management
- the maintenance and evolution of architecture
- architecture traceability
- using architecture approaches in chaotic environments

The SEI will take these requests under advisement.

6 Future of SATURN

The vision for SATURN is to be a forum for a growing number of practitioners to share practices and expertise in what does and does not work. It is open to users of software architecture technology and those interested in promoting software architecture practices. The SEI intends to make SATURN an annual event. Based on the positive feedback from the first and second SATURN workshops in 2005 and 2006, the SEI is planning a third SATURN workshop in Pittsburgh in the spring of 2007.

Participants suggested the following ideas for the next SATURN workshop:

- Broaden the scope of the workshop beyond SEI technology, but still focus on architecture.
- Get more industrial participants (e.g., financial, pharmaceutical).
- Present more demos (during breaks and lunch hours).
- Solicit presentations on sustainable architecture descriptions and show more interconnections among the methods.
- Continue to address system and software architecture relationships.
- Invite system architects to the meeting.
- Solicit presentations on more ATAM case studies and other applications of methods.
- The two-hour format of the working sessions worked well. In the future, improve the preparation for them by having one or two people create a 10-minute vignette.
- Involve standards bodies.
- Provide more modern guidance for life-cycle engineering experience.
- Ask registrants to complete a profile when registering.
- Consider having birds of a feather (BoF) and poster sessions.
- Involve more students who will become software architects.
- Facilitate remote participation (e.g., through Web casts, Web-based dialogs).

For more information and announcements, go to http://www.sei.cmu.edu/architecture/saturn/.

Appendix Acronyms

ACDM
Architecture-Centric Development Method

ADD
Attribute-Driven Design

AGV
Automatic Guided Vehicle

AI
artificial intelligence

ALMA
Architecture-Level Modifiability Analysis

ALPSM
Architecture Level Prediction of Software Maintenance

ARB
Architecture Review Board

ARID
Active Reviews for Intermediate Designs

ARMIN
Architecture Reconstruction and Mining

ATAM
Architecture Tradeoff Analysis Method

BoF
birds of a feather

C4ISR
command, control, communications, computers, intelligence, surveillance, and reconnaissance

C&C
Component-and-Connector

CB
cost benefit

CBAM
Cost Benefit Analysis Method

CCB
change control board

CONOPS
concept of operations

DoD
Department of Defense

DoDAF
DoD Architecture Framework

DoE
Department of Energy

ERP
Enterprise Resource Planning

FFRDC
federally funded research and development center

FRAP
Facilitated Risk Analysis Process

GA
global analysis

GEAR
Good Enough Architectural Requirements

GIS
Geographic Information System

GOMS
Goals, Operators, Methods, and Selection Rules

GSD
Global Software Development

GUI
graphical user interface

IEEE
Institute of Electrical and Electronics Engineers

INCOSE
International Council on Systems Engineering

IV&V
Independent Verification and Validation

J2EE
Java 2 Enterprise Edition

MDRE
Model-Driven Requirements Engineering

NASA
National Aeronautics and Space Administration

OCTAVE
Operationally Critical Threat, Asset, and Vulnerability Evaluation

OMG
Object Management Group

PAAM
Predictive Analysis, Availability Method

PAPM
Predictive Analysis, Performance Method

PASM
Predictive Analysis, Security Method

Q&A
question and answer

QAW
Quality Attribute Workshop

R&D
research and development

RACM
Reused Architectural Component Method

RCAP
Raytheon Certified Architect Program

RE
risk exposure

REAP
Raytheon Enterprise Architecture Process

RMA
Rate Monotonic Analysis

ROI
return on investment

SAAM
Software Architecture Analysis Method

SAT
Software Architecture Technology

SATURN
Software Architecture Technology User Network

SEI
Software Engineering Institute

SEMA
Software Engineering Measurement and Analysis

SOA
service-oriented architecture

SoS
system of systems

TIG
Technical Interest Group

TOGAF
The Open Group Architecture Framework

UML
Unified Modeling Language

V&B
Views and Beyond

XP
Extreme Programming

References

URLs are valid as of the publication date of this document.

[Barbacci 03]
Barbacci, M. R.; Ellison, R.; Lattanze, A. J.; Stafford, J. A.; Weinstock, C. B.; & Wood, W. G. *Quality Attribute Workshops (QAWs), Third Edition* (CMU/SEI-2003-TR-016, ADA418428). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. http://www.sei.cmu.edu/publications/documents/03.reports/03tr016.html.

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

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

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

[IEEE 00]
Institute of Electrical and Electronics Engineers. *Recommended Practice for Architectural Description of Software-Intensive Systems* (IEEE Standard 1471-2000). New York, NY: Institute of Electrical and Electronics Engineers, 2000.

[Kazman 02]
Kazman, R.; O'Brien, L.; & Verhoef, C. *Architecture Reconstruction Guidelines, Third Edition* (CMU/SEI-2002-TR-034, ADA421612). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002. http://www.sei.cmu.edu/publications/documents/02.reports/02tr034.html.

[SEI 05]
Software Engineering Institute. *SEI Software Architecture Technology User Network (SATURN) Workshop 2005*. http://www.sei.cmu.edu/architecture/saturn/saturn_2005.html (2005).

[SEI 06a]
Software Engineering Institute. SEI Software Architecture Technology User Network (SATURN) Workshop. http://www.sei.cmu.edu/architecture/saturn/ (2006).

[SEI 06b]
Software Engineering Institute. Vision, Mission, and Strategy. http://www.sei.cmu.edu/topics/about/vision.html (2006).

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	August 2006	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Proceedings of the Second Software Architecture Technology User Network (SATURN) Workshop	FA8721-05-C-0003

6. AUTHOR(S)

Robert L. Nord

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2006-TR-010

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	ESC-TR-2006-010

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

The second Carnegie Mellon® Software Engineering Institute (SEI) Software Architecture Technology User Network (SATURN) Workshop was held April 25-26, 2006 in Pittsburgh, Pennsylvania. A total of 61 software systems engineers, architects, technical managers, product managers, and researchers exchanged best practices and lessons learned in applying SEI software architecture technology in an architecture-driven development or acquisition project. In the closing session, workshop participants noted these highlights: presentations showing the methods in action, a comparison of multiple SEI Architecture Tradeoff Analysis Method® (ATAM®) evaluations and cross-wise analysis, the workshop format using interactive presentations, a good mix of academic and industry perspectives, and a sharing of workshop results.

This report describes the workshop format, discussion, and results, as well as plans for future SATURN workshops. Key topics covered in the workshop and noted by the participants were the future plans of the SEI's Software Architecture Technology Initiative, the overall integration of software architecture methods and techniques, and the experiences others shared in applying the methods and transitioning them for use. Slides for the presentations and recordings of the keynote talks are available at the SATURN workshop Web site: http://www.sei.cmu.edu/architecture/saturn/.

14. SUBJECT TERMS	15. NUMBER OF PAGES
software architecture, architecture-centric methods, architectural best practices, quality attribute scenario, quality attribute behavior	71

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL