Software Engineering Institute

# A Technical History of the SEI

**Larry Druffel**

**January 2017**

**SPECIAL REPORT**
CMU/SEI-2016-SR-027

**SEI Director's Office**



http://www.sei.cmu.edu

Carnegie Mellon University

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.



This report was prepared for the
SEI Administrative Agent
AFLCMC/PZM
20 Schilling Circle, Bldg 1305, 3rd floor
Hanscom AFB, MA 01731-2125



DM-0003942

# Table of Contents

# List of Figures

# Foreword

**Angel Jordan**
**University Professor Emeritus, Provost Emeritus**
**Carnegie Mellon University**

This report chronicles the technical contributions of the Software Engineering Institute (SEI) since its inception in 1984. It is published at an opportune time, the 30th anniversary of the SEI's formation.

I am writing this foreword as a close observer of the institute from the beginning. Over the years, I have had the opportunity to observe the institute as provost of Carnegie Mellon University (CMU), twice as acting director of the SEI, as University Professor Emeritus, and now, regularly, as just a helper providing advice as needed to division directors and members of the technical staff in their interactions with faculty members on the campus of CMU. The report provides a comprehensive account of the SEI's programs, activities, and initiatives during its existence. It is comprehensive, but not exhaustive. To enumerate and describe in more detail the numerous programs, projects, activities, publications, etc., of an institution that has a relatively long, rich history and a great deal of accomplishments would be a daunting task and beyond the scope of the report.

The introduction gives a crisp description of the SEI, enumerating its key sponsor, its parent institution, and key members of its staff at its inception. It continues with a section describing briefly its formation, its early history, and its evolution over the years as an organization. In this part of the report, the reader will find that changes in the leadership took place several times, with attendant changes in strategies.

In seven chapters, the report chronicles key contributions to software engineering in architecture, education and training, real-time embedded systems, software engineering methods, forensics, management, and security. In the section titled "The Future of Software Engineering," the past three chief technical officers, Bill Scherlis, Doug Schmidt, and Kevin Fall, venture to give their own account of the field in the future. The report is not the end of the story. It is just the first installment. It captures the SEI strategy and also discusses developments outside the SEI.

As described in the introduction, the Software Engineering Institute is a federally funded research and development center (FFRDC) sponsored by the U.S. Department of Defense (DoD), through the Office of the Under Secretary of Defense for Acquisition and Technology at the time, 1984. The SEI contract was competitively awarded to Carnegie Mellon University in December 1984. The DoD established the SEI to advance the practice of software engineering because quality software that is produced on schedule and within budget was deemed to be a critical component of U.S. defense systems. In fact, it was widely believed at the time that there was a crisis in software production and the SEI was to be created as a national resource in software engineering and technology to mitigate that crisis.

As eloquently described, prior to 1984 there were a number of commissions calling for the creation of a Software Engineering Institute funded by the DoD. Members of these commissions came

from the government, industry, and academic institutions prominent in computer science and software engineering. As conceived, the Software Engineering Institute was to be administered by a university and chartered as an FFRDC. A model of an FFRDC, albeit in other technologies, was the Lincoln Laboratory, administered by the Massachusetts Institute of Technology and funded by the DoD. In 1984 there had been no new FFRDCs formed in 20 years.

When the competitive process was established, a number of universities with expertise in software engineering responded with well-conceived proposals. The responding teams from the different institutions were well coordinated, and the communities where those institutions resided orchestrated powerful lobbying campaigns. Carnegie Mellon University did not fall short in attracting political support from its congressional delegation in Pennsylvania and other states with academic institutions allied with Carnegie Mellon University in the quest to attract the SEI.

It was then when, as provost of CMU, I played a role on behalf of the SEI, spearheading the lobbying campaign and leading the team of computer scientists to put together a strong proposal and assembling a meaningful team of computer scientists and technologists who would form the starting nucleus of the SEI. A key part of the proposal was a strong board of visitors, made of prominent people with very strong reputations, from universities (other than CMU), government, and industry. The proposed acting director of the SEI was a respected computer scientist and software engineer who was to head the Department of Computer Science at CMU. After the SEI was launched and a permanent director was found, the acting director was to return to his academic position. Ultimately he became the Dean of the School of Computer Science, formed in 1988 from computer-related disciplines at CMU: the Department of Computer Science, the Robotics Institute, the Language Translation Center, precursor of the current Language Technology Institute, and other related units.

After the SEI was granted to CMU, key members of the institute were hired from inside CMU and other institutions, including industry and government. Programs responding to the charter of the SEI were formed and a number of initiatives were started, including a strong Industrial Affiliates Program and an Educational Program to confer a Master of Science in Software Engineering. This MS in Software Engineering, which served as a model in the country and abroad, was eventually transferred to the Computer Science Department on campus and is now a successful program in the Institute for Software Research (ISR) in the School of Computer Science.

The SEI started its operations in December 1984. The first contract for the first five years, 1985-1990, was funded with $100 million. The contract was renewed in 1990 for another five years with $150 million, and again in 1995 with the same amount. Eventually a permanent director was attracted after a national search. After a year of operations under the leadership of the new director, some difficulties arose owing in part to problems of assimilation and different cultures between the director, its members, and CMU. The director resigned and I, the provost, assumed the acting directorship of the SEI, while continuing to direct the rest of the university, but closely advised by Larry Druffel. After equilibrium was established and the momentum of the SEI was restored, Larry, at the time an executive in a software company in California (who had played a key role in the original conception of the SEI while at the DoD), was attracted as director in 1986. Under Larry, the SEI was propelled to a new trajectory toward becoming a national resource in software engineering as conceived in its charter. He served for 10 years, and moved to another position of leadership in another institution in another part of the country. A few years ago he returned

to the SEI on a part-time basis as a visiting scientist playing an important role as advisor on strategy. For his service to the SEI with distinction in different capacities, including the editorship of this report, Carnegie Mellon owes him great gratitude.

The charter evolved and changed over the years to respond to changes in the environment in accord with the sponsoring organizations, on the watch of the following directors: Larry Druffel (1986-1996), Stephen E. Cross (1996-2002), Angel Jordan (Acting Director in 2003), and Paul Nielsen (2004-to date).

Over the three decades of its existence, the SEI has made many technical contributions to software engineering for which we all can be justifiably proud. The SEI has a track record of leadership in software engineering that has enabled the DoD to take advantage of technology improvements. The report shows clearly that the SEI has not only had impact on specific DoD programs, but has provided leadership to the broader computing community that has brought further development that benefits the DoD. The SEI has also contributed greatly to defining software engineering as a discipline. With its actions and works, it has demonstrated the importance of technology transition. Further, it has demonstrated that having a collection of software experts familiar with research and technology trends, and becoming familiar with DoD needs, can develop technology solutions to satisfy those anticipated needs.

The SEI offers an opportunity that is unique. The DoD challenged the university community to formulate an approach to define the profession of software engineering and offered to fund the response to the challenge. CMU accepted the challenge. What the DoD asks in return is that the SEI focus on those software engineering problem areas that plague the DoD.

Although the SEI is not the lone player in that endeavor, many of our peers recognize that they have a stake in the outcome and are motivated to help, whether out of commitment to the profession, because of a personal desire for recognition, or simply to influence the SEI's thinking along the lines important to that contributor. Consequently, we not only have an interesting and exciting challenge, we have immediate access to leaders in our profession. In adopting the approach to providing leadership in achieving consensus, we thereby encourage the community to participate actively in our endeavors, making them our partners.

## Robert J. Kent
## President of SOSSEC Inc.

In the summer of 1984, I was a program manager in the United States Air Force Electronic System Division at Hanscom Air Force Base in Massachusetts. I was given a directive to establish a DoD Software Engineering Institute to address the perceived software crisis in the development of mission-critical DoD systems. The following was to be the mission:

> *The Software Engineering Institute (SEI) shall provide this: bring the ablest professional minds and the most effective technology to bear on rapid improvement of the quality of operational software in mission-critical computer systems. The Institute shall accelerate the reduction to practice modern software engineering techniques and methods and shall promulgate use of modern techniques and methods to help the mission-critical systems. The Institute shall establish standards of excellence for software engineering practice.*

During 1985, a competitive source selection process was undertaken by the Department of Defense to select a university team to address the mission of the SEI and its establishment as a federally funded research and development center. Seven university teams competed for the award of the DoD's SEI. Of these, Carnegie Mellon University in Pittsburgh, Pennsylvania, emerged as the leading candidate after an exhaustive selection process. CMU's vision for the Institute was considered outstanding and its past history in engineering science was more than expected. CMU was subsequently awarded a five-year, $100 million contract that has renewed every five years up to this date and is going through the renewal process this year.

At the outset, the Institute, which was headquartered on the campus of CMU, had trouble with startup due to various pressures from the DoD, the academic community, and industry. These pressures resulted in the replacement of two directors in the first year of operation, stalling the SEI's effectiveness. At this point, in 1986, Dr. Larry Druffel was selected to be the director of the SEI. Larry brought not only an outstanding background in the science of engineering but a passion for the vision and a leadership style that was able to harmonize the various cultures and agendas surrounding the SEI. Over the following 10 years, Larry was able to revamp the very nature and methods by which the DoD approached software-intensive systems. One of his achievements was to bring focus to the SEI's specific initiatives aimed at improving engineering capabilities. One of these initiatives led to the development of the software maturity model and the process models derived from it. They have stood the test of time and have fundamentally changed the industry. It was Larry's early innovations that set the course for this revolution.

This report details the evolution of the SEI from its inception to today, and how Larry was able to establish the SEI as a national asset in software engineering. His legacy, which now has been passed through three additional generations of capable and visionary leaders, has been lasting and the Institute continues to lead the nation in software engineering under the able guidance of Dr. Paul Nielsen. Larry has continued to be involved in each generation of leadership as a special advisor so his vision continues to set the sense and direction for the SEI. Anyone who has an interest in the evolution of software engineering should read this report.

## Blaise Durante
## Deputy Assistant Secretary for Acquisition Integration for the Air Force

As the SEI develops a history of its technical contributions, I believe it would be helpful for the reader to understand how the SEI has helped the Air Force and other military departments from a user's perspective, rather than simply from a technical perspective. In my role as Deputy Assistant Secretary for Acquisition Integration for the Air Force, I have always been concerned with the Air Force's ability to acquire software-intensive systems responsibly. In general, our acquisition managers are experts in specific application areas such as aeronautics, avionics, unmanned aerial vehicles (UAVs), satellite control, communications systems, and weapons control, but not in software technology. While our acquisition managers may have some knowledge of software, they do not in general have a deep understanding of software development and are most certainly not software engineers. I have worked with the SEI to provide guidance to these program managers. The SEI has helped many individual programs, but the challenge has been to translate the lessons learned from those individual programs to more general guidance applicable across the range of Air Force, and ideally, other DoD applications. In the 10 years since I first challenged the SEI to help these program managers, the SEI has made significant contributions in four general areas.

### Evaluating and Assessing Program Execution

The SEI has developed and employed effective, repeatable methods to evaluate the execution of software-intensive systems acquisition programs, identify problem areas, and provide actionable recommendations for improvement.

*Process In-Execution Reviews (PIERs)* are short-term independent evaluations of processes and practices. The SEI worked with Air Force personnel to develop the PIER method based on the Capability Maturity Model Integration (CMMI) framework (independent of the Standard CMMI Appraisal Method for Process Improvement [SCAMPI]), to provide a rapid, low-cost, targeted approach to evaluating process execution. The SEI has collaborated with program office teams to conduct PIERs during both program execution and competitive down-select to verify that contractors are applying processes and operating consistent with the capability level described in their proposals. PIER has enabled program managers to quickly obtain insight into contractor execution and gain confidence about program cost, schedule, and quality, or to make necessary course corrections before problems reach critical mass. The PIER method has been successfully deployed within multiple Air Force acquisition programs, enabling program managers to take decisive action regarding contractors' technical and management performance.

*Independent Technical Assessments (ITAs)* are independent, comprehensive examinations of programs having difficulty, or needing help with special technical/acquisition areas, and are typically requested by program executive officers (PEOs) and Air Force acquisition officials. ITA teams interview program and contractor staff and analyze acquisition and technical work products to characterize problems, identify root causes, and make recommendations for corrective action. The SEI has, in some cases, supported follow-on efforts to improve system quality/performance and software acquisition practices, and any follow-on assessments requested by the Assistant Secretary of the Air Force (Acquisition) (SAF/AQ). The SEI technical staff has supported numerous ITAs on Air Force programs to enable troubled software-intensive systems acquisitions to get back on the right track.

## Adopting New Technical Approaches for Acquisition

The SEI continually evaluates new technologies and methods for their applicability to DoD acquisition programs, and collaborates with DoD and program leadership to effectively address and appropriately integrate these innovations.

*Agile methods:* In recent years, as Agile methods have matured, DoD contractors have begun building internal Agile capabilities and initiated pilot usage efforts on DoD programs. Formal DoD guidance, templates, and best practices have lagged and are not yet in place to address the incorporation of these methods. The SEI has engaged in extensive research to identify lessons learned across commercial and multiple DoD programs on the application of Agile methods and has especially focused on the regulatory and policy environment surrounding DoD acquisitions. Over the last several years, the SEI team has established a key leadership position in the DoD and contractor community and leveraged this research to deliver technical notes[1,2,3,] educational courses and materials, webinars, conference presentations, and direct program support to DoD programs. The objective of these outreach efforts is to educate acquisition professionals about the integration of Agile methods into programs and the successful management of contractor relationships when these techniques are employed.

*Service-oriented architecture (SOA) and business/acquisition systems:* The SEI worked with the Air Force to demonstrate service-oriented architecture concepts by leading a team of technical experts from several Air Force financial management programs of record in specifying, designing, and prototyping a foundational service: a Program Master Relation Key (PMRK) service. PMRK had the potential to become the authoritative source of the Acquisition Program Master List and correctly link authoritative data from each system into an integrated picture of Air Force acquisitions. This effort demonstrated the positive effects that a "disruptive" technology like SOA could have on acquisition processes and how SOA might be used to resolve the limitations inherent in stove-piped legacy business systems.

## Support to Policy and Leadership Projects

The SEI is a trusted advisor on matters of software-related policy and leadership.

*Technology Readiness Assessments (TRAs):* At the request of SAF/AQ, the SEI has provided team members for independent review teams/panels to support a large number of TRAs on space, command and control (C2), avionics, and weapons systems programs since 2007. In FY08-09, additional SEI staff participated on the TD-1-12 (Air Force Smart Operations–21, Developing and Sustaining Warfighting Systems Core Process) team to develop recommendations on improving the use of TRAs when considering software. The SEI staff involved in these efforts identified the

1 Agile Methods: Selected DoD Management and Acquisition Concerns, http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9769

2 SEI Agile Research Forum: Agile Methods: Agile Methods: Tools, Techniques, and Practices for the DoD Community, http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=21728

3 Considerations for Using Agile in DoD Acquisition, http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9273

synergy between the TRA team's observations of Technology Readiness Levels (TRLs) as currently applied to software, and the TD-1-12 team's efforts at adapting the TRA process to better accommodate software and shared concerns appropriately. The TD-1-12 team's report significantly influenced the software-related content in the July 2009 revision of the TRA Deskbook.

*Cyberspace Task Force:* When the Office of the Chief of Staff of the Air Force (CSAF) and the Office of the Secretary of the Air Force (SECAF) stood up a cyberspace task force to support the provisional Air Force Cyber Command (AFCYBER), the SEI was tasked with aiding the task force's charter to comprehend and integrate Air Force efforts regarding cyberspace. The SEI team developed a series of white papers for task force members on various topics in order to quickly disseminate expertise in the cyber domain. As a part of this effort, the SEI also organized a two-day cyberspace academic training and education workshop that brought Air Force leaders together with cyber curricula experts from universities established in the field. This workshop emphasized operational needs, training and development requirements, and strategic requirements. This overall effort helped set the stage for Air Force Cyber Operations within the acquisition community.

*National Research Council Report:* In FY11, SEI staff contributed to the National Research Council (NRC) Committee on *Examination of the U.S. Air Force's Aircraft Sustainment Needs in the Future and its Strategy to Meet Those Needs*.[4] The committee report identified critical technical, infrastructure, and policy issues (including software acquisition, development, and sustainment) surrounding the sustainment of the Air Force's aging aircraft fleet.

*Policy/Standards Development:* The SEI participates in the Air Force Strategic Software Improvement Program (AFSSIP) Working Group. As a member of this group, the SEI has participated in the development of the first version of the Air Force Software Guidebook, identified challenges in Air Force software sustainment, and provided independent feedback to the Air Force on software-related updates to the Defense Acquisition Guidebook and various Air Force instructions and guidebooks.

### Education and Training for the Acquisition Community

The SEI develops innovative professional development courses and publications relevant to—and used by—all levels of acquisition professionals.

The SEI published *The Adventures of Ricky and Stick: Fables in Software Acquisition* in 2006, and met with a strongly positive result from the acquisition community across the DoD. *Ricky and Stick* presents common acquisition challenges in the form of fables illustrated by comic strips, juxtaposed against the relevant portions of the 5000 series. *Ricky and Stick* remains a popular, though unofficial, source of rich insight (and entertainment) for beginning program managers.

*Software Acquisition Survival Skills (SASS):* The SASS course uses a hands-on approach to introduce program managers and PMO staff to the many specific challenges associated with the acquisition of software-intensive systems. For each software challenge, the course provides background information, ways to recognize problems, avoidance methods, and recovery techniques. SASS has been taught to hundreds of participants at multiple program offices across the U.S. Air Force

4 The NRC report is available for download at http://www.nap.edu/catalog.php?record_id=13177

(USAF) and the rest of the DoD. It has played a vital role in initial training and familiarization with software-specific issues.

*Software Concepts and Issues for Senior Leaders:* As dependence on software increases and systems grow in size and complexity, a significant problem is that many senior acquisition personnel are not aware of the software factors critical to system success and therefore lack the skills to review programs at key milestones and determine that they are executable and on track. This course helps to make senior leaders better aware of tradeoffs and critical success factors for the acquisition of software-intensive systems and to familiarize senior leadership with new/advanced techniques that can pragmatically be applied to their programs. The SEI team originally developed the course to support general officer transition into senior acquisition positions.

## Consequence of SEI Support on Air Force Programs

As a direct result of SEI support for Air Force programs, the Air Force has been more effective in utilizing new techniques and acquiring software-intensive systems. The SEI not only helped the Air Force avoid or overcome significant technical difficulties with many of its programs, such as GPS, F-22, Global Hawk, ALR-69A, IDECS, ABIDES, and B1 FIDL, it has also helped the Air Force develop a cadre of acquisition managers who understand how to manage software-intensive systems. The investments made in the SEI during my tenure have provided significant impact and value to the acquisition community in both the short term (immediate program issues) and for the future in terms of training and education and technology adoption. The SEI has certainly delivered on its role as an FFRDC!

# Preface

This report chronicles the technical accomplishments of the Software Engineering Institute and its impact on the Department of Defense software community, as well as on the broader software engineering community. While it is historical in nature, the report is not a history per se; the focus is on institutional technical accomplishments. While the SEI is nothing without the people who make up the institution, the focus is on technical accomplishments rather than on the people who produced those accomplishments.

Indeed, this report was motivated in part by the passing of two senior members of the SEI staff, both SEI Fellows, within 18 months of one another. With their passing and the retirement of several others came the realization that corporate memory of early accomplishments that led to the current work would forever be lost. This led to the initiation of this project. However, the purpose of this report is to focus on the accomplishments, leaving the discussion of the people involved and their contributions for another time.

The technical accomplishments of the SEI are interwoven with the technical developments in the broader software engineering community. In almost every case, the SEI was influenced by ideas and collaborative participation from industry, university, and government people. In many cases, the broader community was also influenced by SEI activities and developments. Just as software engineering is not driven by a waterfall development model, neither was there a linear interaction between the SEI and the software engineering community. It was, and is, truly interactive. Inside the institution, it is equally difficult to attribute credit. Ideas are proposed in brainstorming sessions, and filtered and refined by members of the staff. Therefore, rather than attribute specific work to individuals, this report focuses on the technical accomplishments while attempting to acknowledge the sources of influence. Attribution will follow the accepted practice of citing references that can be found at the end of each section. Even that device is inadequate because so much of the influence was through workshops and verbal interactions.

This report is intended for readers with varying technical backgrounds whose expertise is not in software engineering but who are interested in the SEI technical contributions and their lineage. Acronyms will be spelled out and technical jargon will be avoided to the extent possible. Software engineers and students will find useful references in understanding how ideas evolved, as long as they understand that the report is necessarily SEI-centric and does not always capture parallel developments elsewhere.

## Organization

The report is organized by technical areas. The technical work and accomplishments are described in the following chapters, which are organized into areas of importance to the mission of the SEI: Real-Time Embedded Systems Engineering, Education and Training, Management, Security, Software Engineering Methods, Architecture, and Forensics. These are not mutually exclusive categories, and, indeed, there are intersections and overlaps.

For each of the these areas a general introduction, supported by a timeline of activities, provides insight into the rationale for the work and the ways in which ideas generated in prior work influenced thinking in later work. Since much of the SEI work was influenced by ideas from outside the SEI, that outside influence will be indicated where it was significant. Failure to recognize outside influence is more a factor of space available on the timelines than repudiation of that influence.

For each major activity shown on the timeline, a brief description will be provided following a similar format:

- A summary of the problem that existed before the SEI engaged
- The idea that was the basis for improvement—where did it come from?
- The consequence—how is the technical environment different because of SEI effort?
- The views of experts outside the SEI who are familiar with the work, where available. Since this report is written by SEI staff, it is useful to have validation by an outside perspective.
- A summary of the SEI contribution—what did the SEI do and what did the SEI not do?

Since this is not the end of the story—only the first volume—a section is included discussing the current SEI technical strategy and offering some insight into trends that are likely to influence the future of software engineering and, therefore, the future of the SEI.

### Suggestions for Readers

*A reader interested in gaining a general understanding* of the SEI and its technical contributions can find that general understanding by reading the introduction and conclusion. The introduction provides the historical context and rationale for the SEI and its technical work. The conclusion summarizes the major technical accomplishments. These two chapters assume little or no software engineering background.

*A reader interested in a more thorough understanding of the breadth of SEI work* could add the introduction to each of the chapters to the general introduction and conclusion. These section introductions summarize the SEI work in seven different technical areas, along with the rationale. While the section introductions assume some familiarity with the field of software engineering, professionals in other fields should find them accessible.

*A reader interested in discussion of SEI work in a specific area,* such as software architecture or security, might skip directly from the general introduction to the appropriate section introductions.

*A reader interested in a more detailed treatment of a specific subject,* such as rate monotonic analysis or undergraduate software engineering curricula, might skip directly to the appropriate subsection. Although these subsections are necessarily of a summary nature, they assume a technical background and provide citations to supporting literature.

However readers choose to read this, we hope they find this historical perspective of the SEI's work and its context interesting and informative.

# Abstract

This report chronicles the technical accomplishments of the Software Engineering Institute and its impact on the Department of Defense software community, as well as on the broader software engineering community. The technical accomplishments of the SEI are interwoven with the technical developments in the broader software engineering community. The described technical work is organized into areas of importance to the mission of the SEI: Real-Time Embedded Systems, Education and Training, Management, Security, Software Engineering Methods, Software Architecture, and Computer Forensics.

# 1 Introduction

# 1.0 Introduction

In December 1984, the Department of Defense (DoD) awarded a contract to Carnegie Mellon University (CMU) to manage a federally funded research and development center (FFRDC) called the Software Engineering Institute (SEI). The contract award was based on a competitive request for proposals (RFP) issued to the university community in May 1984. Seven university or university/industry consortia offered proposals, and Carnegie Mellon University was the successful proposer, in part because of its strong engineering and computer science programs and the "engineering mindset" in its research efforts. The SEI is a university-based FFRDC following the model of Lincoln Labs, which is part of the Massachusetts Institute of Technology. This designation distinguishes the SEI from the systems engineering FFRDCs, such as MITRE and Aerospace Corp., which are free-standing, non-profit corporations.

Responsibility for contract management and technical oversight was assigned to the Air Force Systems Command (AFSC) at Hanscom Air Force Base in Massachusetts. While the contract remained with AFSC, responsibility for technical oversight was moved from AFSC to the Office of the Under Secretary of Defense for Research and Advanced Technology ((OUSD(R&AT)) in 1987, then to the Defense Advanced Research Projects Agency (DARPA) in 1988, and then to OUSD for Acquisition and Technology (A&T), now OUSD for Acquisition, Technology, and Logistics (AT&L), in 1998. These changes brought new perspectives to the oversight and concomitant changes to the strategic imperatives, which are discussed in subsequent paragraphs, along with the context.

## 1.0.1 The DoD Software Environment in 1984 Motivated Formation of the SEI

The proposal to create the SEI originated in 1982 as part of the proposed Software Technology for Adaptable, Reliable Systems (STARS) program—a DoD software initiative developed by a joint task force of DoD software professionals with support from industry [DoD 1983, Druffel 1983]. At the time, DoD leaders realized that software technology was becoming the enabler for flexibility and integration in mission-critical systems, but they also recognized that software was often the cause of system delays and failures. A task force was chartered to develop an understanding of the underlying problems and propose a broad research program for DoD software-reliant systems [DoD 1982].

Although the DoD had traditionally been the leader in the application of computing, that trend reversed dramatically in the 1970s for a variety of reasons, including the difficulty the DoD was experiencing in applying evolving technology [Mowery 1996]. The military departments faced major challenges in developing software for mission-critical systems because a significant component involved managing hardware devices (such as sensors and actuators) and control systems in real time, that is, within the cycle time of the sensor and control mechanisms, often milliseconds or microseconds (now nanoseconds). To meet the efficiency needs of these real-time systems and the need for interfaces to hardware devices, software for DoD systems was developed in assembly language and/or other low-level languages, some of which were specific to a military department or program (e.g., Jovial for the Air Force and CMS-2 for the Navy). To a large extent, programming at these low levels precluded the use of advancing technology and tools. As a result, the DoD experienced increased development, quality assurance, and sustainment costs, and protracted schedules. In addition, low-level code was an inhibitor to the greater levels of integration needed as the DoD began attempting to field larger systems.

In the late 1970s, research demonstrated that compiler technology could produce optimized object code that was efficient enough to handle real-time needs and that higher-level languages could include features to interface with hardware devices. The Deputy Undersecretary of Defense for Research and Advanced Technology ((DUSD (R&AT)) chartered a High Order Language Working Group consisting of representatives from the military departments and DARPA. The working group's charter was to manage the development of a high-level language that would meet the needs of real-time software developers as well as other system developers and that would support the development of a robust software development environment (a collection of supporting tools for developers). DARPA was assigned responsibility for managing the development. This language was eventually called Ada [Carlson 1980].

During the five-year development of the Ada language and supporting environment, specialists with a variety of computer language, compiler, and tools expertise were needed. That expertise was simply not resident in any quantity within the DoD. Likewise, the existing FFRDCs were generally focused on specific system capabilities, such as space, electronics, radar, and logistics. While each had some software expertise, there were too few people with the relevant capability available. The defense industry and university community were willing to assist, but the mechanisms for rapid access to the needed expertise were cumbersome.

DARPA solved this problem by creating a group of distinguished reviewers who were effective in bringing the necessary expertise to the effort. DARPA was able to create such a mechanism because of its broad influence with the computing community at that time. That solution was not available to the military departments, however, and therefore could not be easily replicated; in any case, it would not scale. As the joint task force that produced the STARS proposal began its deliberations, the same lack of available expertise was evident. A large part of the motivation for proposing the SEI was the need to have an organization of software engineers and software researchers familiar with DoD-related problems available to assist the DoD. As such, the SEI was intended to enable the DoD to gain a long-term benefit from the DoD software initiative by continuing to transition evolving technology.

Although the STARS program was well received within DoD and by the defense industry, resistance to the idea of a new FFRDC threatened to delay the entire initiative. The DUSD(R&AT), therefore, pressed forward with the STARS program and chartered a blue ribbon panel to further evaluate the proposed Software Engineering Institute. The panel, which was composed of senior industry and university leaders with support from senior DoD people, returned with a strong recommendation that the DoD establish the SEI [Eastman 1983].

Although the Ada program and the STARS program provided context for initiating the creation of the SEI, the SEI was independent of both programs. Over time, the SEI provided technical support to both efforts, but its operation and direction were clearly separate.

### 1.0.2 SEI Charter—Improve the State of the Practice of Software Engineering

The blue ribbon panel affirmed the joint task force recommendation that the SEI mission have four components: technology transition, research, direct support, and education, with the principal focus on technology transition. The mission statement clearly said, "...The Institute shall accelerate the reduction to practice of modern software engineering techniques and methods and shall

promulgate use of modern techniques and methods throughout the mission-critical systems community." (Subsequent sponsoring agreements shortened the mission statement to "provide leadership in advancing the state of practice of software engineering in support of DoD systems.")

Both the blue ribbon panel and the joint task force recognized that while technology developments from the research community offered promise, those developments needed refinement to make them applicable for practicing engineers, who in turn would need training and further tools support. They also recognized that DoD program managers and their defense industry counterparts were reluctant to adopt a new technology until the supporting infrastructure was in place. That infrastructure included not just training but also support tools offered by commercial companies that would maintain the tools and be available to provide support when needed. The joint task force proposal was for 60 percent of SEI line[5] work to involve technology transition, and noted that this component of the SEI charter was the raison d'etre. When the Deputy Secretary of Defense—who was the source selection authority for the acquisition—received the final briefing from the selection team, the technology transition component of the mission was the deciding factor for him.[6]

The proposal also recognized that to stay abreast of technology, the SEI must be an active participant in the research community. It is not possible to stay abreast of advances in technology and engineering simply by reading the literature. Active participants have their ideas shaped by colleagues. They publish peer-reviewed papers in top conferences and journals, are invited to conferences during which new ideas are tested, are invited to be on program committees where papers are reviewed, and become reviewers for publications where they have the opportunity to see new ideas long before they are published. For these reasons, the proposal was for the SEI to spend 10 percent of its line effort in research.

Likewise, the task force recognized, and the panel affirmed, that an important issue for the DoD and the defense industry was the availability of properly educated software engineers. They recognized that software engineering requires a different set of knowledge and skills than conventional computer science or electrical engineering, the traditional sources of software developers. They proposed that the SEI allocate 10 percent of its line effort to defining and promoting software engineering education.

Finally, the task force recognized that the SEI must have a deep understanding of DoD software issues. This understanding can only be maintained by direct participation in defense systems software development, without competing with defense industry. The proposal was for the SEI to allocate 20 percent of its line effort to direct support.

While the charter clearly stated that the SEI was expected to improve the practice of software engineering on behalf of the mission-critical community, the task force and blue ribbon panel were both clear that unless the SEI accepted the challenge of helping improve the state of the practice in the broader defense and supporting commercial industries, the SEI would not be truly successful. Conversely, while success depended on the SEI's ability to influence the broader software

5 The terms *line* or *core* have been used to refer to work funded by the Congressional line item established by the DoD to provide the base funding for the SEI.

6 Private recollection, Bob Kent, Source Selection Board chairman and first SEI program manager.

community, to do so without benefit to the DoD would clearly be neither appropriate nor sufficient.

### 1.0.3 Software Engineering Context into Which SEI was Formed and Evolved

At the time, even the notion that software development could be called an engineering discipline was debated. There were, after all, few analytic techniques available to a "software engineer;" and there was no set of accepted practices to guide managers, developers, and maintainers of software. There was no widely accepted curriculum for preparing a student for such activities. Few universities offered such a program (most universities did not even offer software engineering courses), and few faculty were prepared to teach the subject. Moreover, there was no accepted body of practice for professionals. Indeed, although the term "software engineering" was coined at a conference in 1968 [Naur 1968], the term was notional and still not yet well defined in 1983.

In the DoD, software development and maintenance was an acknowledged source of failure [DoD 1982]. The department was about to adopt the Ada language for all new developments; but few were familiar with the software engineering concepts that the language enabled, and even fewer were prepared to adopt the new programming paradigm.

An example of the prevalent misconceptions was the assumption that the systems architecture and hardware architecture should be defined before any thought was given to the software. The DoD acquisition and review processes even dictated this approach, based on the assumption that systems engineering could be separated from software engineering and that system capabilities could be defined with no regard to how they might be implemented in software. This assumption held that since software is changeable, software developers could respond appropriately and subsequent changes to the hardware could be accommodated.

This hardware-centric acquisition process assumed that software could be completely defined before any implementation began. It also generally ignored the impact on the software structure of later changes to requirements or hardware characteristics. (The notion of software architecture had not yet surfaced, at least not in the DoD). A consequence, for instance, is that systems would often be fielded with computer hardware that was one or two generations old, real-time systems would deadlock due to timing conflicts, software development would be behind schedule and over budget delaying release of the system, and systems maintenance often led to disastrous results [DoD 1982]. One reaction to these problems yielded the tendency to build custom hardware, which often exacerbated the software problems.

### 1.0.4 The SEI Began by Interpreting the Charter and Developing a Technical Strategy

With this context, CMU began the journey that has taken the SEI into technology that was undefined in 1984 and into arenas that were not—and probably could not have been—predicted. With a small staff recruited from the CMU Computer Science department, the SEI took up temporary residence in an old factory while a new building was constructed. Experts were, thus, recruited to an entity that was unknown, before a new administrative infrastructure was created and, most importantly, a technical strategy envisioned [Barbacci 1985].

### 1.0.5 Interpreting the Charter

In creating a technical strategy, the SEI engaged in some definitional activity that is important to understand in evaluating the technical contributions in this report. Since the term *software engineering* was prominent in the charter, was part of the SEI name, and was to be chiseled in granite on the front of the building, it seemed prudent to define the term. The SEI adapted a definition of engineering, "the application of science and mathematics to produce products..." to derive the definition of software engineering, "the application of science, mathematics, computing, and the application domain to produce software products."

Since *technology transition* was a significant portion of the charter, time was spent in understanding what that entails. The SEI adopted the view that transition implies

- making the technology practical
- providing the education and training materials from which practicing engineers can be prepared to apply the technology
- demonstrating that the technology works in a particular application domain
- making practicing engineers aware of the technology
- ensuring that the infrastructure was established to sustain its use and to continue development

The latter requires that commercial companies develop products to support technology and members of the software community continue its development, thereby enabling the SEI to move on to other work. The SEI understood that just as it requires an order of magnitude more effort and cost to turn a prototype into a product as it did to create the prototype, so the time and effort to transition technology broadly and successfully is substantially greater than the effort to create the technology in the first place [Redwine 1985].

The *research* component of SEI work also needed careful consideration. The research paradigm in the sciences was well understood, but research in software engineering was not as well defined. There were no software engineering PhD programs to use as reference. Indeed, there were few, if any, researchers with computer science PhDs whose dissertations focused on software engineering.

Engineering research is not the same as research in the sciences. Whereas scientific research seeks to prove some scientific principle about the physical world through experimentation and observation, the focus of research in the engineering disciplines is often on finding solutions to difficult problems that can be generalized. Established engineering disciplines, such as electrical, mechanical, and particularly chemical, provided a useful model, which often involved creating a new engineering process, developing a new design method, or proposing mechanisms for increasing efficiency of some process. The SEI concluded that each of these activities was appropriate to software engineering. In addition, the SEI concluded that additional research activities more relevant to software were appropriate, including demonstrating that some new approach would scale and building a prototype that would demonstrate a new approach could work in a particular domain. Such research requires real problems and access to those who encounter those problems. As such, it is not an activity that is easily pursued by an individual or small team because the problem set usually involves large and complex software systems with interfaces that are not tidy and uniform. That is one reason why software engineering research is difficult in a purely academic setting.

Defining the *education* component of the work was more straightforward, although deciding on what activities would best support the mission was a bit harder. It became clear that these activities can easily intersect with technology transition and direct support.

In one sense, defining the *direct support* component of the charter was straightforward because there were models from the system engineering FFRDCs. It was trickier to select specific work for the SEI, however, because it needed to meet at least the following three constraints:

1. not compete with the work of other FFRDCs or with industry
2. further the SEI goals of developing an understanding of DoD needs while supporting the transition work
3. not simply providing bodies to do what others can do

The balance of work was also complicated by the fact that since 20 percent of the SEI line work was to be in direct support, some program offices began to see it as an opportunity to get free help. The Air Force program manager, therefore, began to encourage the SEI to migrate its direct support work from line funding to funding provided by DoD program offices seeking assistance. Such work was guided initially by Technical Objectives and Plans Statements (TO&P)—now called Project Work Statements (PWS)—subject to the original three constraints. Over time, the demand for PWS work has grown significantly, so the original balance envisioned soon became impractical. Likewise, the SEI was encouraged by the SEI program manager to assist other U.S. government agencies under TO&P/PWS funding. Funding from these agencies not only permits the SEI to apply its technical solutions more broadly, but also enables the SEI to develop capabilities that can be applied to the DoD.

The SEI soon realized that these components of the mission were not as cleanly delineated as their definitions might imply, and the percentages soon became notional guidelines. As the demand for TO&P/PWS work increased, the percentage of direct support dominated the mix of work. Moreover, while all TO&P/PWS work was in direct support of the mission of the sponsoring organization, not all fit into the original category of direct support. Indeed, some agencies used TO&P/PWS to sponsor research. The term “direct support” eventually gave way to the more appropriate “technical support.”

### 1.0.6 Evolution of the Effort Composition Was Driven by Experience and Guidance from the Principal Sponsor

The SEI interpretation and evolution of its charter has, and continues to be, conducted in close coordination with the DoD and defense industry. The DoD, which has played an active role under the management of the principal sponsor, also seeks input from industry and academia. For instance, in 1994 DARPA initiated a blue ribbon panel review of the SEI operation, before initiating contract renewal activities, to confirm that the SEI was fulfilling the intended mission and was continuing to evolve to meet DoD needs. The resulting panel report praised the SEI contributions to DoD, endorsed renewal of the contract, and reiterated the need for technology transition [DoD 1994].This comprehensive review process has been continued every five years as part of the contract renewal process.

Following the model of MIT Lincoln Laboratory, oversight of the SEI has been provided by a Joint Advisory Committee (JAC) consisting of senior executives from the Office of the Secretary

of Defense and flag officers from the military departments. This oversight body reviews and approves the strategy and the annual program plan. The JAC is supported by an executive group (JAC/EG), composed of executives representing the JAC organizations. The JAC/EG provides a more in-depth evaluation of the strategic plans and annual work plan, and recommends the appropriate actions to the JAC. While operational management is the responsibility of the SEI director, the sponsoring organization assigns a program manager to ensure that the JAC and JAC/EG guidance is carried out within the provisions of the contract. The original Air Force program manager formed a Technical Advisory Group (TAG) composed of government and academic experts to advise them on technical matters. The TAG has been continued to the present and has become part of the oversight function.

The principal responsibility of the program manager has traditionally involved two primary functions: (1) guiding the SEI's selection of effort to be conducted under line funding, and (2) overseeing efforts conducted under separately funded project work statements (PWS) to ensure that these efforts are consistent with the needs of the DoD and associated U.S. government agencies.

Just as the SEI understanding of its mission has changed and the technology upon which it bases its work has changed, so have the needs of the DoD changed. Furthermore, as responsibility for the sponsoring agent has been shifted from the Air Force to OSD to DARPA and back to OSD, the individual perspectives of those who filled the (program manager/executive agent) position has changed. In 2010, the SEI was directed to concentrate line funding primarily on research and some workforce development, leaving technical support for PWS funding.

Consequently, in 2010, the DoD sponsoring agreement modified the mission statement to provide greater emphasis on research,

> *…to provide technical leadership and innovation through research and development to advance the practice of software engineering and technology in support of DoD needs.*

The SEI's core statement [SEI 2010] specified three areas in which the SEI has traditionally provided value, although the new guidance shifts the emphasis for line-funded work to R&D. The three areas were

**(1) Research and Development**

- research projects that make significant improvements to software engineering and related disciplines
- collaborations that leverage work found in industrial research, academia, and government laboratories
- maintaining cognizance of the global software state of the art/state of the practice in software engineering and related disciplines to identify potential advancements, trends, issues, and new strategic directions for DoD systems

**(2) Technical Support**

- the delivery of technical support addressing specified software engineering problems that impede the government's ability to develop, acquire, deploy, evolve, and sustain high-quality software-reliant systems at a predictable performance, cost, and schedule

**(3) Workforce Development**

- the development of materials that improve the competence of the software engineering workforce.[7] Work in this area includes the development of education and training materials as well as bringing improved practices to the attention of practitioners and managers through workshops, books, webinars, and other effective delivery mechanisms. This activity is performed in conjunction with DoD, government, academic, and industry organizations.

Responding to this new guidance, the SEI balance of activities can be visualized by the following graphic.

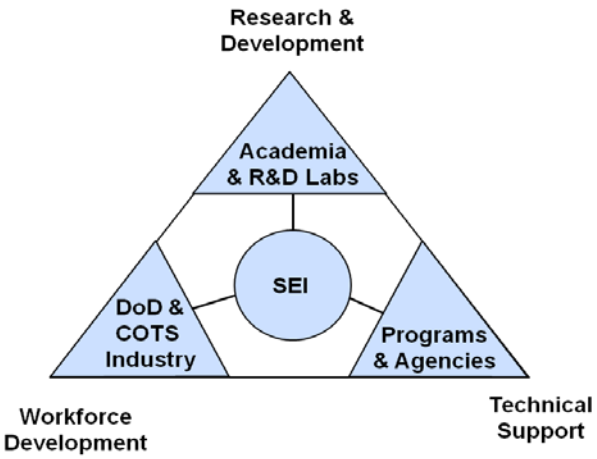


*Figure 1: Balance of SEI Activities*

Experience has demonstrated that technology transition is not a separable activity. Rather it is viewed as a continuum as illustrated in Figure 2.


*Figure 2: Technology Transition Continuum*

The SEI establishes a pipeline of activities in different stages of maturity covering the first four elements of that continuum—Explore, Create, Apply, and Amplify.

- **Explore**: Research projects in this phase are line funded. They investigate new areas of research—areas that are outside the scope of current research plans. Projects in this phase may also build a technical and business case for a more substantial effort to further explore a promising new technical idea.
- **Create**: Research in this phase is aimed at demonstrating the feasibility and potential utility of new ways to enhance software-enabled capabilities. Work in this phase is primarily line funded, although sometimes PWS and Collaborative Research and Development Agreement

7 Research results have no benefit for the DoD unless they help improve the ability of practitioners and managers to avoid or solve software-related problems.

(CRADA) funding is available to explore the potential applicability of the research to problems of software-reliant systems.

- **Apply**: Work in this phase is devoted to applying research results to a variety of systems to validate the results in representative contexts as well as to demonstrate their effectiveness, applicability, and adoptability. Research results are refined and extended in this phase. Work is primarily funded via PWS or CRADA, although some line funding may be used to refine and extend the results based on experience gained in applying the results. In addition, line funding may be used to gather data on costs and benefits to help researchers validate and improve their work.
- **Amplify**: Work in this phase is focused on widespread transition of a research result whose value has been proven in the Apply phase. Activities include organizing workshops and conferences for those interested in applying the result, licensing others to teach practical applications of the result, publishing books, and transitioning further development to other parties. The goal is to foster the development of a self-sustaining infrastructure within which the innovation can flourish without SEI participation.

As described above, line funding is used primarily to support research in the Explore and Create stages, although PWS funding is sometimes available for this kind of research. The later stages are primarily funded with PWS or CRADA funding from organizations seeking to apply new approaches to their software challenges.

### 1.0.7 The SEI Proposes Work Based on an Evolving Technical Strategy

The SEI recognized from the beginning that its work must be driven by a clearly articulated technical strategy. The technical strategy was not, and never has been, formulated solely within the SEI. With the help of defense industry, DoD, and university advisors/partners, the SEI developed a technical strategy to deal not only with the context of the time, but to position the SEI to help the DoD meet future software challenges. Over the years, the SEI technical strategy has evolved with changes in technology and changes in the challenges facing the DoD.

For example, the challenges in the areas of education and training, management, and real-time embedded systems were known early on. In contrast, while the challenges to internet security were predictable, no serious incident had yet been experienced to validate the need for internet security. Other challenges and technology opportunities were not yet understood, or even predicted. From the beginning, the SEI has worked to ensure that its technical strategy evolves to address new challenges and opportunities with the goal of preparing the technology before the DoD experiences the need.

The technical component of the strategy is necessarily based on the state of practice at the time and the technical trends that offer opportunities for improvement. The SEI is a relatively small organization for the breadth of its mission, so an equally important question driving the strategy is how to produce the necessary impact. Thus, an important consideration in formulating the strategy is to identify the points of leverage that will allow the SEI to have impact beyond its size constraints. The SEI strategy evolves based not just on technology but on what the SEI is learning about how to bring about the greatest impact.

Each year, the SEI proposes a work plan based on its rolling five-year technical strategy. Ideas for work projects may come from within the SEI, from practicing engineers in the defense industry, from colleagues in the research community, or from government employees. Ideas often arise while SEI staff are engaged in direct support to specific programs. When the SEI perceives that the problems experienced by a program are representative of a class of problems, or predictive of a class of future problems, the SEI experts seek not only to help the specific program but also to develop solutions that can be broadened and replicated.

Regardless of the source, the SEI measures all new ideas against its vision and evolving strategy and proposes work only when it has the appropriate expertise to carry out the work. The SEI consistently and constantly recruits leaders in new and focused technical areas to ensure that its results will be the best the technology will support. The SEI also recognizes that software engineering requires both expertise in software-related technologies and familiarity with the relevant applications domain. While the SEI staff naturally reflects experience with a broad range of applications, the SEI has consciously chosen not to focus on specific application domains, such as avionics, fire control, or command and control, which would increase the size of SEI staff unnecessarily. Rather, the SEI partners with relevant defense industry and government organizations who have a deep understanding of specific applications.

The SEI's focus on strategy-driven work selection continues. One reason for the consistent impact on the community is that the SEI has articulated a consistent vision that brings credibility over time. The section titled "The Future of Software Engineering" lays out the SEI strategy based on technology trends and defense needs in the foreseeable future. This strategy will guide future work selection.

### 1.0.8 Mechanisms for Engaging the Community

The goal of preparing technology before the DoD experiences the need requires the SEI to develop a deep understanding of DoD systems, culture, acquisition processes, and capabilities. The SEI recognized that such understanding could only be acquired by working on real problems and real systems, but the challenge was to do so without competing with the defense industry. The SEI, therefore, embarked on a partnership approach with industry and DoD program offices. DoD programs sponsored SEI staff to assist prime contractors in applying evolving technology to systems under development.

A second mechanism the SEI has employed to ensure that its work is relevant to the DoD is the resident affiliate program. Software professionals from industry and the DoD are invited to spend up to 18 months at the SEI to work on specific projects. The benefit to the resident affiliate is that he or she participates in the development of the technology and takes that knowledge to his or her home organization. The resident affiliate also helps the SEI project remain grounded on real problems, often even bringing such problems to the project. As of August 2014, 292 government and industry resident affiliates have lent their expertise to the SEI.

The SEI adopted a similar mechanism to access specialized research talent for its efforts by employing visiting scientists on either a full-time or part-time basis. These visiting scientists brought an understanding of evolving technology to complement the SEI internal research activities. About 306 visiting scientists have engaged in SEI work as of August 2014.

Another effective mechanism has been the use of structured workshops. SEI staff often invite leading members of the software engineering community to attend workshops in which evolving technology or problem areas are submitted to intense investigation and debate. All participants benefit from the interaction and are free to take advantage of their newly gained perspective in their own work, while the SEI uses the combined perspective to shape its work.

The SEI has undergone significant change in its first three decades and expects to undergo further changes in the future as the technology changes and the needs of the DoD change. Despite these changes, several characteristics have—and are expected to—remain constant: leadership, innovation, quality, the focus on engineering for software-intensive systems, and impact on its principal sponsor, the DoD.

Each reader is invited to make his or her own judgment about the relative importance and impact of the SEI on DoD systems and on the software engineering community.

Much of the early work will appear dated, overcome by later developments, even mundane by today's standards. Of course, the work must be evaluated in the context of its time. In many cases, the contributions were leadership contributions, many now transitioned to the state of the practice. Thus, while the SEI may not have been the first into an area, it often entered an area that was new—in many cases, an area in which the terms were undefined. The SEI sees its responsibility to collaborate with the software engineering community to help bring some order and direction to the area and to the field. Through these efforts, the SEI has earned a reputation of leadership that encourages collaboration and invites participation by the best software engineers of the day.

### 1.0.9 References

[Barbacci 1985] Barbacci, M. R.; Habermann, A. N.; & Shaw, M. "The Software Engineering Institute: Bridging Practice and Potential." *IEEE Software 2*, 6 (November 1985): 4-21 (ISSN: 0740-7459).

[Carlson 1980] Carlson, M; Druffel, L.; Fisher, D.; & Whitaker, W. "Introducing Ada," 263-271. *ACM '80: Proceedings of the ACM 1980 Annual Conference*. Nashville, TN, October 1980. ACM, 1980.

[DoD 1982] Department of Defense. *Report of the DoD Joint Task Force on Software Problems* (Stock No. ADA123449). National Technical Information Service, 1982. www.dtic.mil/docs/citations/ADA123449

[DoD 1983] Department of Defense. *Software Technology for Adaptable Reliable Systems (STARS) Program Strategy* (Stock No. ADA128981). National Technical Information Service, 1983. www.dtic.mil/dtic/tr/fulltext/u2/a128918.pdf

[DoD 1994] Department of Defense. *Blue Ribbon Comprehensive Review of the Software Engineering Institute, Prepared for the Advanced Research Projects Agency.* DoD, 1994. Not publicly available.

[DoD 2010] Department of Defense. "Annex A: CMU's SEI FFRDC Core Statement." *DoD Sponsoring Agreement for the SEI.* DoD, 2010. Not publicly available.

[Druffel 1983] Druffel, Larry E.; Redwine, Samuel T. Jr.; & Riddle, William E. “The STARS Program: Overview and Rationale.” *IEEE Computer 16*, 11 (November 1983): 21-29.

[Eastman 1983] Eastman, Neil S. *Study Report on the DoD Software Engineering Institute.* Institute for Defense Analysis, 1983.

[Mowery 1996] Mowery, D. C. & Langlois, R. N. “Spinning Off and Spinning On.” *Research Policy 25* (1996): 947-966.

[Naur 1969] Naur P. & Randell, B., eds. *Software Engineering: Report of a Conference Sponsored by the NATO Science Committee*. Garmisch, Germany, October 7-11, 1968. Scientific Affairs Division, NATO, 1969.

[Redwine 1985] Redwine, S. T. & Riddle, W. E. “Software Technology Maturation,” 182-188. *Proceedings of the 8th International Conference on Software Engineering*. IEEE Computer Society Press, 1985.

# 2 Real-Time Embedded Systems Engineering

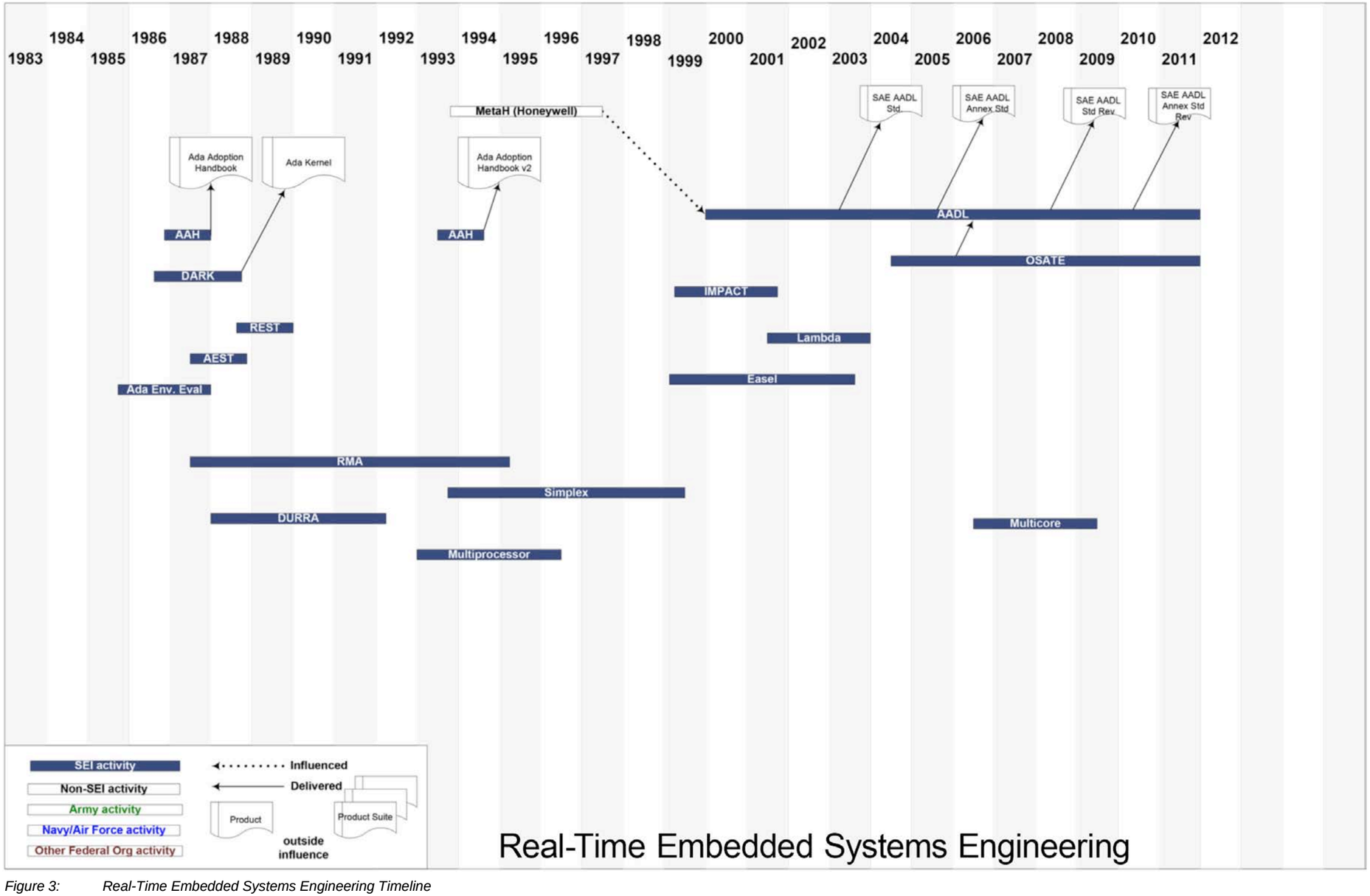


*Figure 3: Real-Time Embedded Systems Engineering Timeline*

## 2.0 Introduction to Real-Time Embedded and Cyber-Physical Systems

Most mission-critical defense systems use embedded processors with hard real-time components in which the software must process inputs from sensors and execute instructions to control devices. Sensor data can arrive synchronously or asynchronously. Each data point is available only for a short period (nano to micro seconds), and the output must be produced fast enough to provide timely control. A terrain-following avionics system is an example: it must process the altimeter, airspeed, and radar inputs in sufficient time to issue altitude controls to follow the terrain. The processor may be performing other lower priority tasks, such as monitoring fuel.

Real-time constraints affect almost all Department of Defense (DoD) systems applications, such as avionics, fire control, vehicle management, missile guidance, radar tracking, and unmanned air vehicle (UAV) control. Indeed, the requirement to manage real-time constraints is one of the factors that distinguishes DoD software from many civilian applications. While civilian applications also exhibit real-time components, the DoD requirements are generally coupled with other factors, including sheer size of the software, security, weapon system safety, and life dependence, that make the real-time component critical [NRC 2010]. As the integration of the processor with the physical system of sensors and controls has become ever tighter, the term *cyber-physical* has been adopted to describe these systems.

### 2.0.1 The SEI Contributed to Early DoD Ada Adoption Effort for Real-Time Embedded Systems

Prior to establishing the SEI, the DoD developed a new programming language called Ada. One of the factors driving development of Ada was the desire to develop real-time software in a higher level language supported by tools to manage the complexity. When the SEI began operating, the DoD had mandated the use of Ada for all mission-critical systems, and DoD program managers had serious concerns regarding early use of the language. Since this was clearly of strategic importance to the DoD and future software-intensive systems, the SEI launched several efforts aimed at addressing the technical and managerial questions about Ada adoption.

A number of commercial Ada compilers were coming to market. While each was required to pass validation by the Ada compiler test suite, the supporting tools making up the software development environment were largely disparate. This was the source of considerable confusion among DoD program managers. The SEI set out to assess the maturity of various compilers and supporting tools to determine their suitability to support DoD programs. For a variety of reasons, DoD programs chose different embedded processors with different instruction set architectures for their systems. Each of those instruction set architectures required different code generators, so that while each could use the same validated compiler and each validated code generator would produce code that correctly executed the Ada instructions, different code generators would have different runtime characteristics.

The SEI established the Ada/Real-Time Embedded Systems Testbed to build the necessary infrastructure to test the runtime performance of code generators. In addition to having the necessary hardware to support testing the suitability of these processors for a given application, the SEI testbed team had the expertise to conduct the testing so that DoD programs did not need to undergo the expense of duplicating that capability. DoD programs that did not choose Ada as their

development language nevertheless needed to test the suitability of their chosen processors and the runtime performance of their chosen compiler. The SEI responded to this need by expanding the Ada Embedded Systems Testbed to the more general case [Weiderman 1989a]. Benchmarks, such as the Hartstone benchmark, were developed that enabled assessment of the performance of a runtime system.

Another source of confusion for DoD program managers concerning the use of Ada for real-time systems were two prevalent notions, motivated in part by historical experience with the use of other high-level languages for real-time systems: (1) the notion that the Ada language needed to be modified to achieve needed real-time solutions, and (2) the practice of extensively modifying the Ada compiler and/or vendor-supplied runtime system. The Distributed Ada Real-time Kernel (DARK) effort was initiated to address two distinct needs of real-time applications: distribution and hard real-time scheduling mechanisms. DARK was a prototype kernel that demonstrated the functionality needed to effectively support the execution of distributed, real-time Ada applications in an embedded computer environment by returning control to the user [Bamberger 1988]. This effort was led by a newly hired SEI staff member who was formerly a member of the Ada Language Design Team and a Distinguished Reviewer, again providing credibility to early SEI work. The resulting prototype was offered to compiler vendors and used by at least one.

These early efforts helped those programs that selected Ada as the development language. But, despite the mandate, that decision was not always an easy one for DoD program managers. A program manager faced many considerations in making the language decision, some driven by myth, some by technology maturity, and some by simple bias. The Air Force program manager for the SEI asked the SEI to develop a factual guide for program managers. Relying on the work of other Ada-related projects at the SEI and the substantial expertise the SEI had accumulated in Ada and real-time systems development, the SEI produced the *Ada Adoption Handbook: A Program Manager's Guide* [Foreman 1987]. The handbook was an objective guide for DoD program managers that addressed many of the myths surrounding the use and practicality of using Ada on defense systems. Its thoroughness and objectivity helped establish the SEI as a source of unbiased guidance on software technology. It was so well received and heavily used in the DoD that the JAC/EG requested an updated version to capture changes in the supporting technology. The updated version was published in 1992 [Hefley 1992].

Adoption of Ada was not as widespread as was originally expected, and the language has since been overtaken by other languages for many applications. To some, this represents failure. That view misses the important point that Ada was a significant step in defining advanced programming languages with capabilities that support the process of software engineering and aid in system reliability, particularly for real-time systems. Ada afforded engineers an opportunity to develop software for real-time embedded systems in a high-level language, including within the language a capability to specify concurrent execution that was formerly available only by making reference to an operating system call. Because the language was well defined and compilers certified as being compliant with the definition, software engineers could develop the software on host machines with powerful software development environments and reliably port the software to a variety of target machines. In essence, Ada helped to perfect the viability of safety critical real-time systems. Several DoD and NASA programs realized these benefits in developing reliable real-time systems. Although some of the people who joined the SEI had been deeply involved in the development of the language and supporting infrastructure, the SEI was not directly involved

in those developments. The SEI was, however, instrumental in making many of the early adopters successful and in providing balanced, unbiased guidance to those faced with the language decision.

### 2.0.2 The SEI Provided an Engineering Basis for Real-Time Systems Development

A major factor in developing real-time software for any language, including assembly language, was the development of the real-time scheduler. There simply was no analytic technique available to ensure that all deadlines could be met. Consequently, practicing engineers relied on experience and gross calculations. When two professors at Carnegie Mellon University (CMU) proved that constraints of an obscure scheduling theory could be relaxed, they approached the SEI for help in making that new theoretic result practical for software. The SEI responded by launching the rate monotonic analysis (RMA) effort. One of the faculty joined the SEI full time and was joined by experienced engineers in the SEI who had Ada and real-time experience. Together they applied the theory, helped develop further modifications, and demonstrated that a previously poorly understood concept of priority inversion could be analytically predicted and prevented. Rate monotonic analysis was quickly transitioned to enable defense industry software engineers to build real-time schedulers that avoid priority inversion and meet all required schedule constraints [Sha 1984]. RMA was applied in the Navy's BSY-1 (Submarine Special Surveillance and Control) Trainer, and the Air Force's F-22. RMA has become state of the practice in developing real-time systems. RMA influenced several standards, including Futurebus, POSIX, real-time CORBA, the Ada language, and the Navy Next Generation Computer Resources (NGCR), and is credited with helping NASA restart the Mars Pathfinder in 1998 after a system shutdown. The transition was so successful that it became the model for effective technology transition at the SEI [Fowler 1993, 1995].

Another complication for DoD real-time embedded systems is that they often have a safety-critical component that must interact with other components. For example, the flight control component in an autopilot is certified to DO178B Level A (the highest level); however, it needs to accept guidance commands from a flight guidance system that is only certified to Level C. Nevertheless, avionics certification requires that Level A software must still function correctly in spite of the software failures in less critical components [DO-178B 1992, RTCA 1992]. The SEI developed an architecture template, called the Simplex architecture, that supports overall safety when a system is composed of both reliable/safe components and less reliable/less safe components [Sha 2001]. This architecture divides a system into two parts: a complex component that cannot be fully verified but is needed to provide important services, and a high-assurance control subsystem that is simple and fully verified. It is designed so that (a) complex components cannot corrupt or interfere with the execution of the high-assurance system, and (b) the data and/or commands from the complex component will not be used unless the resulting system state can be checked in real time that it remains well within the safety and stability envelope. Otherwise, the safety controllers put the system into safety mode. The Simplex architecture also ensures predictable and guaranteed timing behaviors in spite of failures of complex components and provides the ability to restart or replace complex components during operation. Simplex architecture also enables switching the control to alternative components safely. The prototype software was used to demonstrate the concept on an F-16 advanced maneuvering control study using Lockheed Mar-

tin's simulator. Although this prototype software was used for demonstration purposes only, application of the Simplex architecture principles were successfully applied on such systems as the F-22 and F-35.

### 2.0.3 SEI Research Included Software for Parallel Hardware Architectures

Software has traditionally been written in languages that presume a single processor. Although software has been successfully written for parallel machines, the mindset is a radical departure from the single-processor model. Early parallel machines focused largely on applications for which with applications had natural parallelism that could be exploited, such as matrix manipulation and image processing. However, hardware vendors and chip manufacturers recognized that to continue to benefit from the "Moore's Law" curve, they would eventually need to develop general-purpose processors with a high degree of parallelism. The SEI initiated efforts that would enable software engineers to exploit the capabilities of those processors.

Recognizing this future need to support applications running on networks of special-purpose processors executing concurrent tasks, the SEI initiated research in software for heterogeneous machines. This work continued from 1985 through 1992. The heterogeneous machines targeted by this research consisted of general-purpose processors, special-purpose processors, memory boxes, and switches that could be configured in arbitrary logical networks. The application tasks were independent, large-grained, concurrent programs written in various programming languages communicating via message-passing protocols. Heterogeneous machines, such as the one assumed in this research, pushed the leading edge of software engineering [Barbacci 1988]. By 1991, the research focused on improving the practice of developing and maintaining distributed systems. The SEI developed a language and methodology (Durra) ) for implementing distributed, real-time applications on heterogeneous computer systems. The SEI also developed a runtime environment to support distributed applications that use heterogeneous machines.

By 2009, the trend to exploit the advantages of parallelism led to the development of multicore chips, that is, a chip-level multiprocessor (CMP). While these chips offer a significant processing advantage that might be exploited by real-time systems requiring autonomy, such as UAVs, problems occur when threads distributed across multiple processors must synchronize with each other, leading to idle processors and poor utilization. Essentially, there are two aspects that must be considered: (1) allocating and mapping a thread to a processor, and (2) determining the execution order on that processor (i.e., scheduling). A research team composed of SEI staff and CMU professors with extensive experience in scheduling and in practical real-time systems has been addressing multicore scheduling [Andersson 2012a]. It is critical to develop solutions to these problems because the current approach is either to avoid the use of multicore or to turn off all processors except one to use the old sequential solutions. Neither alternatives enables DoD systems to realize the advantages offered by multicore chips.

### 2.0.4 The SEI Developed Analytic Techniques and Supporting Tools for Engineering Real-Time Systems

Modern embedded systems still involve real-time constraints, but as processing capabilities improve, embedded systems are less processor limited in achieving their real-time objectives, even though they are often challenged by added requirements that eat up those spare cycles. In many applications, the result has been a move away from traditional methods toward the use of more

general-purpose techniques but with special care to ensure that real-time constraints are satisfied. For applications such as control systems and autonomous (air, land, or undersea) vehicles, as well as many civilian applications (including medical devices and automobiles), real-time constraints, while still important, are less a primary concern and are, in practice, treated more like other quality attributes. The SEI pursued several efforts that provide engineering analysis to support treating the real-time component as a quality-of-service (QoS) attribute.

Recognizing this trend, the SEI developed a suite of performance reasoning frameworks founded on the principles of generalized rate monotonic analysis (GRMA) for predicting the average and worst-case latency of periodic and stochastic tasks in real-time systems (Lambda-*). The Lambda-* suite can be applied to many different, uniprocessor, real-time systems having a mix of tasks with hard and soft deadlines with periodic and stochastic event inter-arrivals. Some examples include embedded control systems (such as avionics, automotive, and robotic) and multimedia systems (such as audio mixing). Tools were developed to check that a component-based design satisfied various rules imposed by the reasoning framework. This enables the automatic generation of a complete implementation of the design that would exhibit the runtime behavior "predicted" by the reasoning framework, within an explicitly defined confidence interval. The important contribution is that a user would only be able to design or build systems that exhibit predictable behavior by construction, analogous to the way modern programming languages ensure that programs exhibit memory safety by (type system) construction.

The SEI also pursued integrated methods for predictive analytic composition and tradeoff (IMPACT) as a joint effort with CMU faculty and Lockheed Martin Aeronautics Company (LM-Aero). The goal was the development of analytic methods to support the correct temporal composition of systems. The methodology focused on the development of techniques to construct systems having predictable timing performance and composed of pre-analyzed components. Resulting methods included predictable dynamic assembly of software systems from pre-analyzed "software parts" (PAAC), development of temporal analytic composition theory (TACT), predictive models to utilize software and system-level performance measures, and engineering tradeoff analyses involving both runtime attributes and design-time attributes [Saewong 2002].

These methods offered several benefits that support engineering tradeoff analyses at design time and at runtime, including design uniformity using architectural patterns, reduction in rework through system-level analysis conducted at design time, and the ability to address more complex systems by leveraging pre-analysis of architectural patterns. These methods were used on the F-22 embedded avionics simulation to show that all temporal design characteristics expressed in the F-22 challenge problem could be readily modeled and analyzed using a combination of real-time queuing theory (RTQT) and generalized rate monotonic analysis techniques. Furthermore, it allowed LM-Aero and CMU to propose a large-scale DASADA II experiment centered on upgrading the F-22 mission computer. Results and insights from this experiment aimed to reduce both new development and application rehost costs. The team was invited by the U.S. Army Aviation and Missile Command (AMCOM) to propose a large-scale experiment centered on application of technologies to the Sikorsky Black Hawk helicopter.

In the early 1990s, as recognition of the importance of software architecture grew, the SEI sought ways to apply these emerging principles to real-time systems. A DARPA-funded effort that fostered the creation of architecture description languages (ADLs) produced a design at Honeywell

Technology Center, called MetaH, specifically focused on embedded software and supporting RMA [Vestal 1993]. After its successful use on a missile guidance system for the Army, the Avionics Systems Division of SAE International embarked on the development of an international standard. Recognizing its mission to accelerate transition of technology to practice, the SEI agreed to assume leadership in this effort and the evolution of supporting technology. With support of the community, the SEI led acceptance of the SAE Avionics Architecture Description Language (AADL) standard in 2004, with revisions in 2009. The SAE AADL was specifically designed to support modeling and analysis of large-scale embedded software system architectures in terms of an application runtime architecture bound to a computer platform architecture and interacting with a physical system in which it is embedded. The architecture is expressed through concepts with well-defined semantics, such as periodic and aperiodic tasks with sampled and queued communication operating as partitioned system on synchronous or asynchronous networked computer hardware. Standardized extensions to AADL address embedded architecture standards such as ARINC653, analysis of nonfunctional properties such as safety and reliability, as well as architecture-focused requirements capture, validation, and verification. With the release of the standard, the SEI provided an Eclipse-based open source implementation of a tool environment for AADL called OSATE (Open Source AADL Tool Environment) to encourage pilot projects. The SEI continues to be involved in the SAE AADL committee and to work with the aerospace industry, as well as other industry sectors, to foster use of this model-based architecture-centric practice into military systems.

The SEI also developed an agent-based programming language (Easel). This language allowed independent specification of the time characteristics of each system constituent and interactions among the constituents, but without explicit user-level management of the timing interactions. Easel was used for a variety of embedded systems applications, including cooperative UAV control applications.

### 2.0.5 SEI Contributions to Standards

An important factor in the transition of technology to general practice is the existence of a national or international standard. System developers of DoD systems rely heavily on standards to ensure that the infrastructure is present to support use of a particular technology. The SEI has actively influenced real-time standards to provide confidence on the part of defense systems developers in evolving technology. For instance, generalized rate monotonic analysis influenced several standards, including Futurebus, IEEE POSIX, real-time CORBA, the Ada language, and the Navy Next Generation Computer Resources (NGCR). Another example is the IEEE 1003 standard (POSIX). The SEI led the development of the SAE Avionics Architecture Description Language. The SEI also initiated and provided leadership to the IEEE 1003.21 standard, Real-Time Distributed System Communication. The standard includes operations for initialization, asynchronous operations, event management, buffer management, endpoint management, directory services, destination-based message transfer, broadcast, multicast services, labeled messages, and connection management and termination. The standard is defined as a language-independent standard (LIS). That is, a full semantics of the application interface has been defined independent of a particular programming language, allowing the LIS to be bound to multiple programming languages. The standard also includes an extensive formal specification that was completed in 2002.

### 2.0.6 Summary

DoD systems have traditionally been challenged by the real-time needs of embedded systems. Often, the requirements constrained the software architecture and challenged the developers' innovation. From its inception, the SEI has not only addressed the then-current needs of the DoD, but has anticipated its future needs. Through a combination of research into new theory, evolution of analytic techniques and supporting tools, application to real systems, and influence on standards, the SEI has provided the leadership that has enabled defense systems to realize the benefits of integrating systems of embedded processors with hard real-time constraints. In the process, the SEI efforts have matured the practice of software engineering so that one can reason about the behavior of a system and its properties.

### 2.0.7 References

[Andersson 2012a] Andersson, Bjorn; Chaki, Sagar; de Niz, Dionisio; Daugherty, Brian; Kegley, Russell; & White, Jules. "Non-Preemptive Scheduling with History-Dependent Execution Time," 363-372. *Proceedings of the 24th Euromicro Technical Committee Conference on Real-Time Systems (ECRTS)*. Pisa, Italy, July 11-13, 2012. IEEE, 2012. http://www.computer.org/csdl/proceedings/ecrts/2012/4739/00/index.html

[Bamberger 1988] Bamberger, Judith; Coddington, Timothy; Firth, Robert; Klein, Daniel; Stinchcomb, David; Van Scoy, Roger; & Colket, Currie. *Distributed Ada Real-Time Kernel* (CMU/SEI-88-TR-017). Software Engineering Institute, Carnegie Mellon University, 1987. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=10661

[Barbacci 1988] Barbacci, M. R.; Weinstock, C. B.; & Wing, J. M. "Programming at the Processor-Memory-Switch Level," 19-28. *Proceedings of the 10th International Conference on Software Engineering (ICSE)*. Singapore, April 11-15, 1988.

[DO-178B 1992] "Software Considerations in Airborne Systems and Equipment Certification." RTCA/DO-178B, December 1, 1992.

[Doubleday 1992] Doubleday, Dennis & Barbacci, Mario. *Durra: A Task Description Language User's Manual (Version 2)* (CMU/SEI-92-TR-036). Software Engineering Institute, Carnegie Mellon University, 1992. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=11761

[Foreman 1987] Foreman, John & Goodenough, John. *Ada Adoption Handbook: A Program Manager's Guide* (CMU/SEI-87-TR-009). Software Engineering Institute, Carnegie Mellon University, 1987. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=10261

[Fowler 1993] Fowler, Priscilla & Levine, Linda. *Technology Transition Push: A Case Study of Rate Monotonic Analysis (Part 1)* (CMU/SEI-93-TR-029). Software Engineering Institute, Carnegie Mellon University, 1993. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=11993

[Fowler 1995] Fowler, Priscilla & Levine, Linda. *Technology Transition Pull: A Case Study of Rate Monotonic Analysis (Part 2)* (CMU/SEI-93-TR-030). Software Engineering Institute, Carnegie Mellon University, 1995. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=11999

[Hefley1992] Hefley, William; Foreman, John; Engle, Jr., Chuck; & Goodenough, John. *Ada Adoption Handbook: A Program Manager's Guide, Version 2.0* (CMU/SEI-92-TR-029). Software Engineering Institute, Carnegie Mellon University, 1992. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=11721

[NRC 2010] NRC Committee for Advancing Software-Intensive Systems Producibility. *Critical Code*. National Academies Press, 2010 (ISBN-13:978-0-309-15948-7).

[RTCA 1992] RTCA Inc. "Software Considerations in Airborne Systems and Equipment Certification." RTCA/DO-178B, December 1, 1992.

[Saewong 2002] Saewong, S.; Rajkumar, R.; Lehoczky, J. P.; & Klein, M. H. "Analysis of Hierarchical Fixed-Priority Scheduling." *Proceedings of Euromicro Conference on Real-Time Systems.* Vienna, Austria, June 19-21, 2002; 19-21. IEEE, 2002. http://www.computer.org/csdl/proceedings/ecrts/2002/1665/00/16650173-abs.html

[Sha 1984] Sha, Lui & Goodenough, John. "Generalized Rate-Monotonic Scheduling Theory: A Framework for Developing Real-Time Systems." *IEEE Proceedings 82*, 1 (January 1984): 68-82.

[Sha 2001] Sha, L. "Using Simplicity to Control Complexity." *IEEE Software* (July/August 2001): 20-28. IEEE, 2001.

[Weiderman 1989a] Weiderman, Nelson. *Ada Adoption Handbook: Compiler Evaluation and Selection Version 1.0* (CMU/SEI-89-TR-013). Software Engineering Institute, Carnegie Mellon University, 1989. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=10929

[Vestal 1993] Vestal, S. & Binns, P. "Scheduling and Communication in MetaH," 194-200. *Proceedings of the Real-Time Systems Symposium. (RSTS).* Durham, NC, December 1-3, 1993. IEEE, 1993.

## 2.1 Ada/Real-Time Embedded Systems Testbed

### 2.1.1 The Challenge: Evaluating Runtime Performance on Embedded Processors

How real-time systems would be programmed in a high-level language like Ada was just one aspect of Ada adoption for the DoD. Two other important aspects concerned (1) the performance of the code generated by Ada compilers for the various embedded processors used by the DoD and (2) the efficiency of the services provided by the Ada runtime environment. The runtime environment provided services such as process management, storage management, and exception handling for supporting the execution of Ada programs. Prior to the adoption of Ada, such services had been provided either by the application programmer or by a small real-time executive.

There was concern both inside and outside the DoD that about whether Ada could support these real-time needs efficiently. In particular, the semantics of the tasking model and the processing overhead associated with task interactions and context switching were viewed as impediments to the real-time performance demanded of mission-critical software. An SEI report describes the issues and summarizes some of the significant early investigative work of organizations such as the Ada Runtime Environment Working Group (ARTEWG)—a special interest group established by the ACM in 1985—and the Evaluation and Validation team of the DoD's Ada Joint Program Office [Weiderman 1987a].

### 2.1.2 A Solution: A Testbed for Real-Time Performance Evaluation

Any assessment of Ada for mission-critical computing on embedded processors would have to take into account the quality of both the generated code and the runtime execution environment. The SEI established the Ada Embedded Systems Testbed (AEST) in 1986 to investigate these questions. The objective was to generate and disseminate quantitative evaluations of a representative set of vendors' Ada implementations targeted to various embedded processors. The investigations used a test suite of Ada programs comprising existing Ada benchmarks, a simulated real-time application based on a Navy shipboard inertial navigation system (INS), and a new benchmark created specifically for the project.

Criteria for constructing the testbed included requirements for each tested compiler (for example, the smallest value of the pre-defined "Duration" type should be less than 100 microseconds) and its runtime system (for example, the overhead for a context switch should be less than 200 microseconds) [Weiderman 1987b]. An investigation of existing benchmarks led to the selection of the University of Michigan Ada benchmarks [Clapp 1986] and the ACM Special Interest Group on Ada (SIGAda) Performance Issues Working Group (PIWG) benchmarks as the initial test suites [Donohoe 1987].

The testbed itself was a host-target environment in which a cluster of Digital Equipment Corp. Microvax II host machines were connected to a set of target single-board computers with processors, such as a 20 MHz Motorola MC68020, a 16 MHz Intel i80386, and a 15 MHz Fairchild 1750A (with a MIL-STD-1750A instruction set architecture). The Ada cross-compilers for the target boards came from DDC-I, Systems Designers, Tartan Laboratories, TeleSoft, and Verdix. The testbed also included a logic analyzer because one of its construction criteria was the requirement for hardware verification of software timing results. The effort soon evolved beyond the objective of evaluating Ada and was broadened into the Real-Time Embedded Systems Testbed (REST).

Running the initial benchmark suites on the testbed yielded timing data for individual language features (e.g., subroutine calls, task activation, and exception handling), for collections of language features (matrix multiplication and fast Fourier transform), and for runtime environment features (task scheduling and memory management). They also provided insights into the limitations of the benchmarks themselves [Altman 1987a, 1987b]. These insights, coupled with the need to investigate runtime features not supported by the initial benchmarks, led to the creation of the Hartstone benchmark.

Hartstone is a synthetic benchmark for hard real-time applications [Weiderman 1989b]. "Hard" real-time applications *must* meet their specified execution deadlines, as opposed to "soft" real-time applications, where a statistical distribution of response times is acceptable. The "stone" part of the Hartstone name comes from the influence of two important synthetic benchmarks, Whetsone [Curnow 1976] and Dhrystone [Weicker 1984]. Hartstone was created specifically to address deadline-driven computing in Ada, something that currently available benchmarks did not address. The benchmark mimics a real-time application by requiring the completion of a synthetic workload, distributed among several concurrent tasks, within a specified time period.

In parallel with the Hartstone experiments, the testbed also used the simulated INS to collect empirical data on the use of Ada in a time-critical application [Meyers 1988]. In addition to functioning as a composite benchmark, the INS provided an artifact for investigating runtime system support for alternative scheduling policies and the use of Ada in distributed environments. The INS work benefited greatly from having a resident affiliate at the SEI from the U.S. Navy, who contributed a real INS specification, and one from the Australian Department of Defence, who helped adapt the specification to meet the design criteria for a composite benchmark based on the INS.

As DoD programs began choosing other high-level languages such as C, the SEI was asked to support evaluation of the runtime performance of compilers for those languages. The testbed was expanded to satisfy this need for a broader range of real-time system performance issues.

### 2.1.3 The Consequence: Empirical Results for Runtime Performance Hypotheses

The testbed demonstrated the need to validate vendors' performance claims with careful experimentation. The initial benchmark runs showed that (a) numbers must be interpreted in light of the specific configurations and parameter settings established for the tests, and (b) even carefully constructed benchmarks have limitations that can produce anomalous results.

Running Hartstone revealed the wide variation in the timing behavior of various Ada runtime systems. The timing resolution of the system-provided clock varied by three orders of magnitude, even for different compilers running on the same target. The timing resolution and behavior of the Ada "delay" statement (granularity, overall accuracy, and accuracy near zero) were highly correlated with the timing behavior of the system clock. Both had a significant impact on the Hartstone results. Hartstone stress testing also exposed bugs in the runtime environment of several Ada compilers. These were usually manifested as missed deadlines by high-priority, high-frequency Hartstone tasks [Donohoe 1990].

The testbed team worked with several compiler vendors to resolve these issues. The team also collaborated with the creator of the original Whetstone benchmark [Curnow 1976] at the National Physical Laboratory in the United Kingdom to refine the testbed's Ada version of the Whetsone

benchmark. Results of testbed experiments were documented in technical reports and papers and presented at DoD and industry workshops.

Both Hartstone and the INS simulator contributed important artifacts and results to two other SEI efforts: the Real-Time Scheduling in Ada (RTSIA) effort and the Distributed Ada Real-time Kernel (DARK). Both benefited from early testbed experimentation with a programmable real-time clock device driver and the design and test of approaches to periodic task scheduling. Members of the testbed team also collaborated with the Advanced Real-Time (ART) project at Carnegie Mellon University. Several DoD programs based language and processor decisions on the runtime evaluations provided through this testbed.

### 2.1.4 The SEI Contribution

The testbed validated the earlier benchmarking contributions of organizations such as PIWG and the University of Michigan. It also contributed a new benchmark, Hartstone, to address a gap in the area of measuring deadline-driven computing. Hartstone provides a highly parameterized benchmark capable of stress testing Ada runtime systems by varying the workload, priority, frequency, and number of concurrent tasks to be executed. At the time the REST project concluded, organizations other than the SEI were proposing to create a distributed version of Hartstone and to implement it in programming languages other than Ada. The lessons learned from the testbed experiments were incorporated into a comprehensive guide to the selection and evaluation of Ada compilers as a companion to the Ada Adoption Handbook [Weiderman 1989a].

### 2.1.5 References

[Altman 1987a] Altman, Neal & Weiderman, Nelson. *Timing Variation in Dual Loop Benchmarks* (CMU/SEI-87-TR-021). Software Engineering Institute, Carnegie Mellon University, 1987. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=10335

[Altman 1987b] Altman, Neal. *Factors Causing Unexpected Variations in Ada Benchmarks* (CMU/SEI-87-TR-022). Software Engineering Institute, Carnegie Mellon University, 1987. http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=10341

[Clapp 1986] Clapp, Russell M.; Duchesneau, Louis; Vols, Richard A.; Mudge, Trevor N.; & Schultze, T. “Toward Real-Time Performance Benchmarks for Ada.” *Communications of the ACM 29*, 8 (August 1986): 760-778.

[Curnow 1976] Curnow, H. J. & Wichmann, B. A. “A Synthetic Benchmark.” *The Computer Journal 19*, 1 (February 1976): 43-49.

[Donohoe 1987] Donohoe, Patrick. *A Survey of Real-Time Performance Benchmarks for the Ada Programming Language* (CMU/SEI-87-TR-028). Software Engineering Institute, Carnegie Mellon University, 1987. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=10381

[Donohoe 1990] Donohoe, Patrick. *Hartstone Benchmark Results and Analysis* (CMU/SEI-90-TR-007). Software Engineering Institute, Carnegie Mellon University, 1990. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=11177

[Meyers 1988] Weiderman, Nelson & Meyers, B. *Functional Performance Specification for an Inertial Navigation System* (CMU/SEI-88-TR-023). Software Engineering Institute, Carnegie Mellon University, 1988. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=10707

[Weicker 1984] Weicker, R. P. "Dhrystone: A Synthetic Systems Programming Benchmark." *Communications of the ACM 27,* 10 (October 1984):1013-1030.

[Weiderman 1987a] Weiderman, Nelson; Borger, Mark; Cappellini, Andrea; Dart, Susan; Klein, Mark; & Landherr, Stefan. *Ada for Embedded Systems: Issues and Questions* (CMU/SEI-87-TR-026). Software Engineering Institute, Carnegie Mellon University, 1987. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=10363

[Weiderman 1987b] Weiderman, Nelson. *Criteria for Constructing and Using an Ada Embedded System Testbed* (CMU/SEI-87-TR-030). Software Engineering Institute, Carnegie Mellon University, 1987. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=10389

[Weiderman 1989a] Weiderman, Nelson. *Ada Adoption Handbook: Compiler Evaluation and Selection Version 1.0* (CMU/SEI-89-TR-013). Software Engineering Institute, Carnegie Mellon University, 1989. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=10929

[Weiderman 1989b] Weiderman, Nelson. *Hartstone: Synthetic Benchmark Requirements for Hard Real-Time Applications* (CMU/SEI-89-TR-023). Software Engineering Institute, Carnegie Mellon University, 1989. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=10995

## 2.2 Distributed Ada Real-Time Kernel

### 2.2.1 The Challenge: Provide Consistent Support for Ada in Real-Time Systems

As DoD programs began to seriously consider the use of Ada for real-time embedded systems, some developers were not satisfied with Ada language features supporting distributed applications for real-time systems. They wanted the flexibility to make modifications.

There were two proposals offered. The first was for an application-specific tailoring of runtime environments and the addition of compiler-specific pragmas to enhance the real-time capabilities of the language. The consequence of this approach would have meant that the implementation would be compiler-dependent, thereby defeating one of the purposes of a common language, namely, portability. The second proposal was for additional language features in the language. This proposal was also problematic. The language was already defined, and a new round of language definitions would delay its implementation. Potential users in the DoD needed an immediate solution.

### 2.2.2 A Solution: A Distributed Ada Real-Time Kernel

Members of the SEI staff were convinced that applications engineers needed language functionality, not language features [Firth 1987]. They argued that certain areas of functionality were above and beyond the scope of any language, including Ada. They set out to show that it would be possible to leave the decisions about runtime to the applications software and systems engineers who understood the intricacies of the systems they were developing. Their view supported the Ada Joint Program Office perspective that the application-specific runtime library should be considered an integral part of the application, not part of the compiler, and set out to use the Ada package concept as the means of handling distribution and real-time scheduling.

The SEI team built a prototype kernel and made it available for others to use. The kernel communications model consisted of a set of primitives that could be thought of as an underlying set of primitives connected by data paths, following the ISO reference model [Tenenbaum 1981, Zimmerman 1980]. Using this model, the target hardware, the physical layer, and the kernel then implemented the data link, network, and transport layers. The transport layer was visible to the applications development team, while the data link and network layers were encapsulated (hidden) by the transport layer. The applications code then would implement the session, presentation, and applications layers. Errors in a lower layer were reported by the transport layer.

The kernel could then be implemented on each processor in the distributed system. As a result, when developing a process, the software engineer need not know where the other processes would be located, whether on a single processor or across multiple processes. The kernel communication primitives would be used for all inter-process communication [Bamberger 1988].

### 2.2.3 The Consequence: A Prototype Demonstration

The prototype kernel was made available for others to tailor for their applications. It provided a tangible demonstration that the Ada language did not need new features and that it was capable of supporting distributed real-time applications. The Boeing Co. teamed with Wichita State University to successfully port it to a Motorola 68000-based system [Tomayko 1990]. Although no specific system is known to have used the specific code, it nevertheless offered confidence that Ada

could be used successfully in the applications for which it was intended. One compiler vendor adopted the kernel as the basis for its runtime approach.

### 2.2.4 The SEI Contribution

The SEI developed this prototype in parallel with other, complementary, efforts. Compiler vendors were using the Ada package concept to provide primitives for the applications engineers to manage their distributed, real-time applications. Indeed, this approach became the norm. By making the code of the kernel widely available, the SEI demonstrated that Ada could be used for its intended purpose and provided confidence to the early adopters that such an approach was not only feasible, but would be supportable.

### 2.2.5 References

[Bamberger 1988] Bamberger, Judith; Coddington, Timothy; Firth, Robert; Klein, Daniel; Stinchcomb, David; Van Scoy, Roger; & Colket, Currie. *Distributed Ada Real-Time Kernel* (CMU/SEI-88-TR-017). Software Engineering Institute, Carnegie Mellon University, 1987. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=10661

[Firth 1987] Firth, Robert. "A Pragmatic Approach to Ada Insertion," 24-25. *Proceedings of the International Workshop on Real-Time Ada Issues*, Devon, England, May 13-15, 1987.

[Tenenbaum 1981] Tenenbaum, A. S. "Network Protocols." *Computing Surveys 13,* 4 (December 1981): 453-459.

[Tomayko 1990] Tomayko, James & Smith, Brian. *Experiences Porting the Distributed Ada Real-Time Kernel* (CMU/SEI-90-TR-017). Software Engineering Institute, Carnegie Mellon University, 1990. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=11211

[Zimmerman 1980] Zimmerman, H. "OSI Reference Model—The ISO Model of Architecture for Open Systems Interconnections." *IEEE Transactions on Communications COM-28,* 2 (February 1980): 425-432.

## 2.3 Ada Adoption Handbook

### 2.3.1 The Challenge: Where and When to Adopt Use of the Ada Language

When the DoD mandated the use of the Ada language, compilers, runtime systems, and the supporting programming environments were just coming to market. While they held the promise of making software development for real-time systems more effective, there was considerable uncertainty on the part of DoD program managers who had to make difficult decisions about the technical maturity and the production quality of the language support as applied to their system developments. In some cases, although compilers and supporting environments were available, code generators for the embedded processors were still in development and the runtime systems for those processors unproven. As a result, there remained those who were skeptical that a higher order language could be used for real-time embedded systems. While there were some early success stories and considerable pressure on the DoD program managers, there was also a considerable amount of uncertainty, often based on bias or myth rather than hard data.

### 2.3.2 A Solution: The Ada Adoption Handbook

The SEI had committed to helping those DoD program managers who chose to use Ada to do so successfully. Efforts such as the Ada Embedded System Testbed were initiated to provide program managers with the kind of data that allowed them to assess the maturity of the language support for specific embedded processors. While the testbed was under development, compiler maturity for specific processors was just one of many questions that remained unanswered. In some cases, program managers were avoiding the decision or leaning toward the use of other high-level languages because of the lack of authoritative data (of any sort) and the lack of "honest broker" guidance. The Air Force program manager for the SEI asked that the SEI develop definitive guidance upon which DoD program managers could base an informed decision.

By that time, the SEI had several members of the technical staff with a great deal of experience with Ada and with real-time systems. Nevertheless, the requested effort was daunting since there was so much difference in opinion in the software development community between those who understood the promise of Ada and those who were adamantly opposed to its use. The SEI committed to providing an honest assessment, providing the pros and cons of adopting the language, independent of DoD mandate.

The SEI embarked on a strategy that has been employed on many of its efforts. Relying on its internal expertise to separate fact from fiction, the SEI invited comment from those with experience with the language, whether positive or negative. The handbook authors organized this information and supplemented it with their own experience to produce a first draft that was distributed widely for comment. Comments on this first draft were addressed to produce a second draft, which was distributed for additional comment. Subsequent drafts followed the same process. Recognizing that the SEI was dealing with a moving target in that the products to support Ada were maturing at an accelerating rate, the SEI and the Air Force agreed that it was in everyone's best interest to distribute the final version widely to DoD program managers in May 1987 [Foreman 1987]. As the technology supporting Ada changed, the information in the document became obsolete. Therefore, based on requests from a broad constituency, the SEI produced an updated version of the handbook in 1992 [Hefley 1992].

In the handbook, significant emphasis was placed on providing information and suggesting methods that would help program and project managers succeed in using Ada across a broad range of application domains. Although the issues were complex, they were not all unique to Ada. Many of the issues addressed in the handbook must be addressed when developing any software-intensive system in any programming language. The handbook focused on the following topics: program management issues, including costs and technical and program control; Ada's goals and benefits; software tools, with emphasis on compiler validation and quality issues; the state of Ada technology as it related to system engineering; the application of special-purpose languages; issues related to mixing Ada with other languages; possible productivity benefits resulting from software reuse; and implications for education and training.

### 2.3.3 The Consequence: Unbiased Guidance

The handbook provided unbiased guidance upon which to make decisions about the use of Ada. Myths were debunked, misinformation clarified, and some product claims put in context. More than 6,000 copies were distributed, and the handbook became the most widely read SEI publication to that point. The neutrality and technical validity of the treatment helped establish the SEI as a trusted source of information—the "honest broker" reputation that is acknowledged and respected by the software engineering community and that the SEI continues to foster.

**The View from Others**

Despite the fact that some Ada proponents felt the handbook was too neutral with respect to promoting the language and some Ada opponents felt it promoted Ada, no one disagreed with the accuracy of the content. DoD program managers reported that the information provided a basis for making decisions, for asking important questions of their contractors, for justifying and allocating funds for development, and, in some cases, for taking alternative approaches. Prime contractors likewise reported that it provided them the insight necessary to evaluate alternatives when selecting the technology on which to base proposals.

### 2.3.4 The SEI Contribution

As with many other efforts in which the SEI has taken the lead, much of the information in the handbook was gathered from both the DoD and the industry making up the DoD software supply chain. In addition to leadership and organization, the SEI had the expertise to assess the credibility of the information, separating fact from fiction. The SEI recognizes that many senior people in both industry and government invested their time to influence the content and ensure its correctness. Nevertheless, the SEI was responsible for writing the document and accurately portraying the state of Ada and supporting technology at the time the handbook was published.

### 2.3.5 References

[Foreman 1987] Foreman, John & Goodenough, John. *Ada Adoption Handbook: A Program Manager's Guide* (CMU/SEI-87-TR-009). Software Engineering Institute, Carnegie Mellon University, 1987. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=10261

[Hefley 1992] Hefley, William; Foreman, John; Engle, Chuck Jr.; & Goodenough, John. *Ada Adoption Handbook: A Program Manager's Guide, Version 2.0* (CMU/SEI-92-TR-029). Software Engineering Institute, Carnegie Mellon University, 1992. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=11721

## 2.4 Rate Monotonic Analysis

### 2.4.1 The Challenge: Predicting Real-Time Systems' Ability to Meet Performance Deadlines

The majority of defense systems are real-time systems with hard performance deadlines. A major challenge in designing a real-time system is the ability to predict, before the system is built, whether the real-time deadlines will be met at runtime. This is roughly analogous to being able to predict whether a bridge will stand before it is built. While structural engineers have the mathematical tools to make such predictions, no such analytical tool was available for software engineers designing real-time systems. Consequently, real-time behavior was difficult to predict. Most designed systems were using cyclical executives with manually developed timelines. Even engineers with significant prior experience had their systems fail, often catastrophically. As noted in the 1992 National Research Council report *Computing the Future*, "The traditional method of scheduling concurrent tasks is to lay out a timeline manually. A change might require the undo of an entire timeline" [NRC 1992].

Adding to this risk was growing pressure from the operating community to acquire integrated systems, those in which the individual sensor inputs and controls are integrated into a single computer or suite of computers. The F-22 aspiration for integrated avionics is one example [AFSAB 1988]. By the late 1980s, the traditional timeline approach was no longer able to handle the complexity of real-time systems. It was time for a new approach, one based on solid theory and supported by analytic tools.

### 2.4.2 A Solution: Rate Monotonic Analysis

In 1988, a faculty member and his graduate student at Carnegie Mellon University were working under a contract from IBM Federal Systems Division to investigate a way of determining, in advance, whether a fiber optical network could meet all the real-time communication deadlines. They found two theories from a 1973 paper [Liu 1973] proving that if a set of independent periodic tasks has a worst-case processor utilization of less than 69 percent, then all the tasks' jobs will meet their deadlines, provided that higher priorities are given to tasks with higher rates. This was called rate monotonic scheduling (RMS). RMS provided the basis for solving IBM's problem, but IBM's problem also highlighted the practical limitations of RMS. The same paper also proved that, for a set of independent periodic tasks, if priority is assigned to jobs instead of to tasks, using the earliest (job's) deadline-first (EDF) algorithm, then all jobs' deadlines will be met as long as the worst-case processor utilization is less than 100 percent. This paper seemed to provide the underpinnings for a potentially practical theory for designing real-time systems. Naturally, the CMU researchers first tried to extend EDF to address various practical concerns such as task interaction.

Fortunately, the researchers were also given a set of challenging example applications. They quickly determined that with EDF, the problem of maintaining system stability under transient overload does not have low-complexity practical solutions. They also noticed that the 69 percent bound is irrelevant in practice. First, practical control tasks form rate groups, and the schedulability is over 90 percent instead of 69 percent. Second, given a set of periodic tasks that is constrained by the 69 percent bound, these tasks can be easily transformed to achieve much higher

processor utilization. The sample applications allowed them to make the critical decision to build upon RMS instead of EDF. The result of their work was rate monotonic analysis (RMA).

RMA is the application of generalized rate monotonic scheduling [Sha 1984]. It provides the theoretic basis to bring engineering analysis to the design of real-time applications. It requires much less information than the timeline approach and makes it much easier to accommodate integration and evolution of complex real-time systems. RMA also provides the theoretic basis to bring engineering analysis to real-time computing standards, such as languages, operating systems, middleware, and hardware bus arbitration. Experience in applying RMA to real systems motivated the SEI and collaborators to evolve new analytic tools.

### 2.4.3 The Consequence: Engineering Replaces Art

An important factor in RMA is the ability to minimize priority inversion, where a high-priority task is blocked by a lower priority task. It helps system designers predict whether task deadlines will be met before costly implementation. This important factor has been instrumental in enabling RMA to influence a host of hardware and software standards.

Today, RMA is a basic component in real-time computing textbooks and taught in many universities, such as CMU and University of Illinois Urbana-Champaign. A companion RMA handbook provides the definitive guide for practitioners [Klein 1993]. RMA is also the only real-time scheduling technology approved by the Federal Aviation Administration for Level A avionic software in networked control applications with distributed computers, sensors, and actuators. In other practical applications, the F-16 was the first Air Force aircraft that utilized generalized rate monotonic scheduling. In 2000, Lockheed Martin included RMS scheduling in the F-35 design baseline, as it had become an established, foundational engineering practice.

**The View from Others**

*The navigation payload software for the next block of Global Positioning System upgrade recently completed testing. ... This design would have been difficult or impossible prior to the development of rate monotonic theory.*

– L. Doyle, and J. Elzey ITT, Aerospace Communication Division (p.1) [Doyle 1993]

*Through the development of Rate Monotonic Scheduling, we now have a system that will allow [Space Station] Freedom's computers to budget their time, to choose between a variety of tasks, and decide not only which one to do first but how much time to spend in the process.*

– Aaron Cohen, Deputy Administrator of NASA, in an October 1992 lecture (p.3) [Cohen 1992]

New Challenges: A fundamental assumption of real-time scheduling theories, including RMA, is that the worst-case execution time of a task is the same whether it runs alone or with other tasks. Processor cache memory invalidates this assumption. Current multicore architectures exacerbate this problem, because software running in one core could cause severe delays in other cores via the interference of shared last-level cache among cores. Just as RMA has changed many hardware and software standards in the past, RMA offers promise that this multicore design problem will also be fixed in the future. Currently, the University of Illinois at Urbana-Champaign is collaborating with SEI and industry to address this new challenge.

### 2.4.4 The SEI Contribution

The SEI was instrumental in the development of the rate monotonic scheduling paradigm, and its technical staff played a crucial role in the development of the theory. The SEI connected researchers with the user community to ensure that the theory would be relevant to practice. RMA transformed real-time computing practice, and the SEI was instrumental in broadly transitioning the theory.

The SEI's participation in the development of generalized RMS began in the late 1980s, when a team was formed that involved the collaboration between the SEI, industry, and other departments of CMU. This collaboration kept the basic research focused on generating the knowledge for solving high-impact, recurrent real-time computing challenges. The team provided consulting support to early adopters of this technology; and the rapid acceptance by industry enabled the SEI to work with the standards committees to change related open standards on real-time computing so that RMS would be supported consistently. The standards include IEEE Futurebus+ and all the later bus arbitration standards, POSIX real-time extension (real-time OS standard), Ada, real-time CORBA (middleware standard), and real-time Java, among others.

The RMA team addressed real-time computing challenges in the real-world evolving systems, including how many priority levels should be used in hardware and software standards; how to handle the task interaction in a way that maximizes schedulability; and how to integrate the scheduling between aperiodic requests and periodic tasks. This practical work guided the basic research to focus on difficult problems that matter most in practice. The SEI created training workshops, consultation support for early adopters, and a handbook for practitioners [Klein 1993].

The RMA work serves as an excellent example of the SEI role in conducting research that is inspired by real-world needs and ultimately improves the practice of software engineering. As noted by IEEE in promoting an SEI staff member to IEEE Fellow in 1998, the contribution "enabled the transformation of real-time computing practice from an ad hoc process to an engineering process based on analytic methods." [UI 2014].

**The View from Others**

*When was the last time you saw a room of people cheer a group of computer science theorists for their significant practical contribution to advancing human knowledge? :-) It was quite a moment.*

– Dr. Michael Jones, reporting after RMA enabled the rescue of Mars Pathfinder when it encountered real-time computing problems on Mars [Jones 1997]

The 1992 National Academy of Science report, *Computing the Future*, described the generalized rate monotonic scheduling theory as "a major accomplishment in computer science" [NRC 1992].

The DoD saw generalized RMS as "a major payoff," and made this declaration in its 1991 *Software Technology Strategy* (pp. 8-15) [DoD 1991]

### 2.4.5 References

[AFSAB 1988] Air Force Science Advisory Board (SAB). *Integrated Avionics*. SAB, 1988.

[Cohen 1992] Cohen, Aaron. "Charting The Future: Challenges and Promises Ahead of Space Exploration" (Lecture). University of Illinois, Urbana-Champaign, October 28, 1992. Available through https://www.coursehero.com/file/9421167/lect17rmaexactsol/

[DoD 1991] Department of Defense. *Software Technology Strategy*. DoD, 1991.

[Doyle 1993] Doyle, L. & Elzey, J. *Successful Use of Rate Monotonic Theory on a Formidable Real-Time System*. (Technical report). ITT, Aerospace Communication Division, 1993.

[Jones 1997] Jones, Mike. "What Really Happened on Mars Pathfinder." *The Risks Digest 19,* 49 [December 7, 1997]. http://catless.ncl.ac.uk/Risks/19.49.html#subj1

[Klein 1993] Klein, Mark. H.; Ralya, Thomas; Pollak, Bill; Obenza, Ray; & Harbour, Michael Gonzales. *A Practitioner's Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems.* Kluwer Academic Publishers, 1993. (ISBN 0-792393619).

[Liu 1973] Liu, C. L. & Layland, J. W. "Scheduling Algorithm for Multiprogramming in a Hard Real-Time Environment." *Journal of the ACM 20*, 1 (1973): 46-61.

[NRC 1992] National Research Council. *Computing the Future: A Broader Agenda for Computer Science and Engineering*, Selected Accomplishment Section (p. 193). National Academy Press, 1992.

[Sha 1984] Sha, Lui & Goodenough, John. "Generalized Rate-Monotonic Scheduling Theory: A Framework for Developing Real-Time Systems." *IEEE Proceedings 82*, 1 (January 1984): 68-82.

[UI 2014] University of Illinois. *Cyber Physical Systems Integration Lab.* http://publish.illinois.edu/cpsintegrationlab/about/history/ (2014).

## 2.5 Simplex Architecture

### 2.5.1 The Challenge: Ensuring the Integrity of Safety-Critical Systems

Organizations rarely have the resources and technologies to make software systems flawless. Frequently, designers also have to deal with the fact that less than 15 percent of a system's code is newly created for a particular project. Much of the legacy code has known and unknown defects. It boils down to having different grades of code in practical systems: different components come with different levels of quality and with different price tags.

Furthermore, safety-critical components cannot be fully isolated. These components need to interact with less reliable and even unsafe components safely. For example, the flight control component in an autopilot is certified to DO178B Level A (the highest level). However, it needs to accept guidance commands from a flight guidance system that is only certified to Level C. Nevertheless, avionics certification requires that Level A software must still function correctly in spite of the software failures in less critical components [RCTA 1992]. In medical systems, patient-controlled analgesia (PCA) is commonly used after a major surgery. When a patient pushes a button, morphine sulfate (or similar drug) will flow into the patient's blood stream via the infusion pump. Morphine overdoses can be fatal and, indeed, there were fatal accidents in early generations of PCAs.

In this example, the PCA controller is a safety-critical component, but it needs to take commands from a patient whose actions must be assumed to be unsafe. In the first example, the Level A flight controller has to take guidance commands from the less reliable Level C flight guidance subsystem.

### 2.5.2 A Solution: The Simplex Architecture

The SEI developed an architecture template, called the Simplex architecture, that supports the overall safety of a system that is composed of both reliable/safe components and less reliable/less safe components [Sha 2001].

Under the Simplex architecture, a system is divided into two parts: a complex component that cannot be fully verified but is needed to provide important service, and a high-assurance control subsystem that is simple and fully verified. A Simplex architecture is designed in such a way that (1) complex components cannot corrupt or interfere with the execution of the high-assurance system and (2) the data and/or commands from the complex component will not be used unless the resulting system state can be checked in real time that it is remaining well within the safety and stability envelope. Otherwise, the safety controllers put the system into safety mode.[8]

The relation between complex and simple safety controllers must be verified as well formed in the Simplex architecture. This means that the safety controller can use the service of the complex components but the system safety does not depend on the correct functioning of complex components. In short, the key principle of Simplex architecture is to use, but not depend on, the service of complex or legacy components whose correctness cannot be fully verified. This principle is

8 In automated flight control, this means letting the pilot take over. In PCA, this means stopping the morphine and alerting nurses.

also known as *using simplicity to control complexity*. A key factor in applying this principle is to have a simple computation of the bounds to be expected from the more complex component. In the case of the PCA example, the simple bounds might be based on known allowable quantities of morphine over a specific period, factored by the patient's weight. As long as the more complex system (the human) does not exceed those allowable bounds, the human can employ varying amounts of morphine to combat experienced pain, with each patient using his or her own pain tolerance as a guiding factor (the complex component).

The Simplex architecture also ensures predictable and guaranteed timing behaviors in spite of failures of complex components and allows restarting or replacing complex components during operation. Simplex architecture also enables switching the control to alternative components safely. This can be done automatically so that if one complex controller fails, a second, alternative controller using different algorithms can be invoked. An important side effect of this feature is that developers can incrementally compile a new complex controller and switch control to that controller while the system is running.

### 2.5.3 The Consequence: Increased Reliability of Safety-Critical Systems

The architecture principles have been applied successfully to many defense programs as well as commercial systems. Notable applications of Simplex architecture principles include the F-22 and F-35. In acknowledging support from the SEI and the University of Illinois at Urbana-Champaign support during the implementation of those systems, DoD leadership clearly shows that the technology has been highly regarded.

### 2.5.4 The SEI Contribution

The Simplex architecture is a software-fault-tolerant architecture. Prior to its development, the dominant software-fault-tolerant approach was N-version programming [Lyu 1995]. It was shown that the Simplex architecture significantly outperforms N-version programming under a wide range of conditions [Sha 2001].

The Simplex architecture grew out of research at the SEI, and three prototype systems were developed to demonstrate application of the concept. They include an inverted pendulum for experimental purposes, a diving control system funded by the Navy, and an F-16 advanced maneuvering control study using Lockheed Martin's simulator and funded by the Air Force. Important applications and extensions include the support of safety engineering in networked medical device interoperability (sponsored by the National Institutes of Health and led by Massachusetts General Hospital) and its applications to enhance the security of electric power networks (led by the SEI).

Recent extensions and development of technological advances include the System Simplex architecture, in which the safe controller is implemented in field programmable gate arrays (FPGA) [Bak 2009]. System Simplex is robust against operating system failures and security attacks on the application processor. A more economical variant of System Simplex architecture is to implement the safety controller on a secured core in a multicore chip. Currently, the SEI and the University of Illinois Urbana-Champaign are collaborating on the extension of the System Simplex architecture to secure a power generation and distribution network. Another extension is the support of networked control systems, in which the stability enveloped accounts for the implications of distributed control challenges [Yao 2013]. An ongoing research project in extending Simplex is

to integrate Simplex architecture with an L1-adaptive controller designed to tolerate mechanical failures. The resulting technology is called L1-Simplex, which is designed to tolerate concurrent software and mechanical failures under a known fault model.

### 2.5.5 References

[Bak 2009] Bak, S.; Chivukula, D.; Adekunle, O.; Sun, M.; Caccamo, M.; & Sha, L. "The System-Level Simplex Architecture for Improved Real-Time Embedded System Safety." *Proceedings of the 15th IEEE Real-Time and Embedded Technology and Applications Symposium.* RTAS 2009, San Francisco, CA, April 13-16, 2009. IEEE, 2009.

[Lyu 1995] Lyu, Michael R, editor. *Software Fault Tolerance*. John Wiley & Sons, 1995 (ISBN 0-471958888). http://www.cse.cuhk.edu.hk/~lyu/book/sft/

[RCTA 1992] RCTA Inc. *Software Considerations in Airborne Systems and Equipment Certification*. (DO-178B), December 1, 1992.

[Sha 2001] Sha, L. "Using Simplicity to Control Complexity." *IEEE Software 18*, 4 (July/August 2001): 20-28.

[Yao 2013] Yao, J.; Liu, X.; Zhu, G.; & Sha, L. "NetSimplex: Controller Fault Tolerance Architecture in Networked Control Systems." *IEEE Transactions on Industrial Informatics 9*, 1 (February 2013): 346-256.

## 2.6 Software for Heterogeneous Machines

### 2.6.1 The Challenge: Meeting Performance Goals for Real-Time Applications Involving Heterogeneous Machines

Around the time that the SEI was formed, DARPA was sponsoring research, under the Strategic Computing Initiative, in a number of computation-intensive, real-time applications, such as autonomous land vehicles. These applications required processing data obtained from sensors (for example, TV cameras, radar, and sonar), extracting basic features of the terrain, consulting low-level knowledge sources to build hypotheses about paths and obstacles, and consulting higher level knowledge sources to make decisions about current vehicle location, desired target location, and ways to arrive there.

Given the computer technology of the day, the demands for computing cycles were so great that conventional processors could not meet all the performance goals. Certain tasks in the applications required special-purpose processors capable of executing some tasks very quickly—but such processors were perhaps not very useful for other tasks.

### 2.6.2 A Solution: Software for Heterogeneous Machines (Durra)

Recognizing this future need to support applications running on networks of these special-purpose processors executing concurrent tasks, the SEI initiated research in software for heterogeneous machines. This work continued from 1985 through 1992. The heterogeneous machines targeted by this research consisted of general-purpose processors, special-purpose processors, memory boxes, and switches that could be configured in arbitrary logical networks. The application tasks were independent, large-grained, concurrent programs, written in various programming languages and communicating via message passing protocols. Heterogeneous machines, such as the one assumed in this research, pushed the leading edge of software engineering [Barbacci 1988].

By 1991, the research focused on improving the practice of developing and maintaining distributed systems. The SEI had developed a language and methodology (Durra)[9] for implementing distributed, real-time applications on heterogeneous computer systems [Barbacci 1986a, 1997, 1987; Doubleday 1992]. The SEI also developed a runtime environment to support distributed applications that use heterogeneous machines [Weinstock 1989].

This research improved the state of the practice of software engineering by integrating techniques for specifying the software structure of applications, specifying reusable component programs, and specifying the timing and functional behavior of component programs and applications [Barbacci 1986b].

Much of the research was done in collaboration with the CMU Department of Computer Science (now part of the School of Computer Science), building on and leveraging its work on Nectar [Arnould 1989], a prototype heterogeneous machine that was also funded by DARPA to develop applications for an autonomous land vehicle.

9 The name *Durra* is not an acronym. Rather it came from Sorghum bicolor, commonly called sorghum and also known as durra, jowari, or mil—a grass species cultivated for its edible grain. Since the project was dealing with large-grained parallelism, the name seemed appropriate.

### 2.6.3 The Consequence: Successful Demonstration in Prototype Systems

As is often the case with software research conducted before the hardware is fielded that would make the software capabilities necessary for real-world systems, Durra was useful primarily in demonstrations. Those demonstrations gave software engineers the conceptual framework for later application of the principles involved.

The Durra language and runtime technologies were used to support a demonstration project at TRW Defense Systems Group. Specifically, the technology was demonstrated in the context of a command, control, communications, and intelligence (C3I) application developed by TRW, implementing a node using reusable components [Barbacci 1989]. The experiment illustrated the development of a typical Durra application.

The Institute for Simulation and Training at the University of Central Florida used the Durra language and methodology as a tool for analyzing network configurations and computer-generated vehicles in a distributed simulation and training application. Durra aided them in addressing problems related to multiple protocols, multiple levels of fidelity, and multiple technologies used throughout their simulations.

The Hughes Aircraft Co. used the technology to help evaluate the real-time performance of an architecture prior to implementation. The performance of a highly parallel architecture depends upon the match of the algorithm to the hardware. A system designer needs to address questions of the form, “Given an algorithm, on which architecture would it run most efficiently?” or “Given an architecture and an algorithm, what can be done to improve system performance?”

### 2.6.4 The SEI Contribution

The Durra language was an early example of an architecture description language (ADL). It demonstrated the usefulness and possibility of meaningful *application programming*, what we would now call programming at the subsystem and system system-of-systems level. It showed how programming at the application level can be used for real-time analysis. In addition, it introduced important concepts, such as the separation of application structure from behavior and how to deal with configuration and fault-tolerance issues at this level. Durra was not the only such language being developed at the time. (Conic is an example; concepts in Conic influenced Durra and, in turn, Durra influenced Conic [Magee 1989].) There has since been a long line of such languages, including UML, Meta-H [Vestal 1993], and AADL. Each of these languages was, to some extent, influenced by concepts from Durra.

### 2.6.5 References

[Arnould 1989] Arnould E.; Bitz, F.; Cooper, E.; Kung, H. T.; Sansom R.; & Steenkiste, P. “The Design of Nectar: A Network Backplane for Heterogeneous Multicomputers,” 205-206. *ASPLOS-III Proceedings - Third International Conference on Architectural Support for Programming Languages and Operating Systems*., Boston, MA, April 3-6, 1989. ACM, 1989.

[Barbacci 1986a] Barbacci, Mario. *Durra: A Task-Level Description Language Preliminary Reference Manual* (CMU/SEI-86-TR-003). Software Engineering Institute, Carnegie Mellon University, 1986. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=10153

[Barbacci 1986b] Barbacci, Mario & Wing, Jeannette. S*pecifying Functional and Timing Behavior for Real-Time Applications* (CMU/SEI-86-TR-004). Software Engineering Institute, Carnegie Mellon University, 1986. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=10157

[Barbacci 1987] Barbacci, Mario R.; Weinstock, Charles B.; & Wing, Jeanette M. "Durra: Language Support for Large-Grained Parallelism," 56-57. *Proceedings from the Second Workshop on Large-Grained Parallelism.* Hidden Valley, PA, October 11-14, 1987. Carnegie-Mellon University, 1987.

[Barbacci 1988] Barbacci, M. R.; Weinstock, C. B.; & Wing, J. M. "Programming at the Processor-Memory-Switch Level," pp. 11-15. *Proceedings of the 10th International Conference on Software Engineerin*g *(ICSE).* Singapore, April 11-15, 1988. IEEE, 1988.

[Barbacci 1989] Barbacci, Mario; Doubleday, Dennis; & Weinstock, Charles. *Command, Control, Communications, and Intelligence Node: A Durra Application Example* (CMU/SEI-89-TR-009). Software Engineering Institute, Carnegie Mellon University, 1989. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=10905

[Barbacci 1993] Barbacci, M. R.; Weinstock, C. B.; Doubleday, D. L.; Gardner, M. J.; & Lichota, R. W. "Durra: a Structure Description Language for Developing Distributed Applications." *Software Engineering Journal 8,* 2 (March 1993): 83-94. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=206966&isnumber=5294

[Doubleday 1992] Doubleday, Dennis & Barbacci, Mario. *Durra: A Task Description Language User's Manual (Version 2)* (CMU/SEI-92-TR-036). Software Engineering Institute, Carnegie Mellon University, 1992. http://www.sei.cmu.edu/library/abstracts/reports/92tr036.cfm

[Magee 1989] Magee, J.; Kramer, J.; & Sloman, M. "Constructing Distributed Systems in Conic." *IEEE Transactions on Software Engineering 15,* 6 (June 1989).

[Vestal 1993] Vestal, S. & Binns, P. "Scheduling and Communication in MetaH," 194-200. *Proceedings of the Real-Time Systems Symposium (RTSS*). Durham, NC, December 1-3, 1993. IEEE, 1993.

[Weinstock 1989] Weinstock, Charles. *Performance and Reliability Enhancement of the Durra Runtime Environment* (CMU/SEI-89-TR-008). Software Engineering Institute, Carnegie Mellon University, 1989. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=10901

## 2.7 Real-Time Multicore Scheduling

### 2.7.1 The Challenge: Taking Advantage of Multicore Chips

The trend to increase processing power has shifted from increasing the frequency of execution to multiplying the number of processors embedded in a single chip. This trend is known as multicore chips or chip-level multiprocessor (CMP). The increase in processing power is generated by increasing the number of instructions that can be executed concurrently rather than by reducing the time to execute a single instruction. Consequently, an application experiences a speedup in its execution only if it has enough instructions that can be executed concurrently—*parallelizable instructions*. The additional processor capacity made available by having additional cores in a multicore processor can be exploited only if enough parallel instructions can be found in the application. Unfortunately, this limitation is misaligned with the sequential programming model prevailing in current software development practice, where application code is generally developed as a single sequence of instructions under the assumption that they will not execute in parallel.

In many real-time systems, a fair amount of parallelization has already been exploited, but still more is needed to cope with the growing demands of computation imposed on defense systems, such as that imposed by the increased demand for autonomy in unmanned aerial vehicles (UAVs). Today, real-time systems and applications have already been developed using threads that are later scheduled with well-established schedulers to achieve predictable timing behavior. However, the bulk of the scheduling research assumed a single core; and while multiprocessors (not multicore) are already being used and analyzed, such systems are not yet fully understood. For instance, there are cases of anomalies where a system with $N$ processors can miss deadlines with a workload that is only just enough to fill one processor [Dhall 1978]. Similar problems occur when threads distributed across multiple processors need to synchronize with each other, leading to idle processors and poor utilization. Essentially, there are two aspects that must be considered: (1) allocating and mapping a thread to a processor and (2) determining the execution order on that processor; that is, scheduling. The solution to these problems will very likely also involve a change in the structure and in the abstractions used to develop these systems.

### 2.7.2 A Solution: Real-Time Scheduling for Multicore Processors

In 2009, the SEI began to investigate real-time scheduling for multicore processors with a focus on analyzing the problems of task-to-core allocation, synchronization, and the relationship between synchronization and task allocation. The focus was soon extended to analyze variations of multicore processors that include graphical processor units (GPUs). While GPUs are typically used to render graphics, they are often used for general parallel computation as well.

Previous work on scheduling resulted in an increase of the global scheduling utilization to 33 percent for periodic tasks and 50 percent for aperiodic tasks [Andersson 2001, 2003]. Some other approaches chose to use quantized assignments of processor cycles to tasks with a scheduler that calculates a scheduling window at fixed intervals [Srinivasan 2001, Anderson 2006]. However, none of these efforts took into account the task interactions and different tasks and application structures. The SEI partnered with Carnegie Mellon research faculty to explore the combination of task scheduling and task synchronization, creating a coordinated allocation and synchronization algorithm that can obtain up to twice the utilization of non-coordinated ones [Lakshmanan 2009].

The increase in processing capacity offered by multicore processors has enabled smaller form factors that are highly relevant for defense systems, most notably UAVs. However, this trend also brings two more challenges: the mixture of functionality of different criticality levels in the same system and the dependency of the execution time of these functions on environmental conditions.

A system that contains functionality from different criticality levels is known as a mixed-criticality system. In these systems, it is essential to ensure that low-critical tasks do not prevent high-critical ones from meeting their timing requirements (deadlines). The SEI developed a family of scheduling [de Niz 2009], allocation [Lakshmanan 2010], and synchronization [Lakshmanan 2011] algorithms, known as *zero-slack scheduling algorithms*, that implement what is known as asymmetric temporal protection. Asymmetric temporal protection ensures that lower criticality tasks cannot interfere with a higher criticality task, but a higher criticality task can steal CPU cycles from lower criticality tasks, if needed. This asymmetric protection enabled the SEI to also address the second issue of small-form-factor systems, namely, the variability of execution time. This variability is common in autonomous algorithms. For instance, the execution time of collision avoidance algorithms depends on the number of objects encountered in the environment. The asymmetric protection of the zero-slack scheduler allows engineers to double-book processor cycles between high- and low- criticality tasks. This enables high-critical tasks to execute for a long time when extreme environmental conditions occur (such as a large number of obstacles to avoid) by completely stopping lower criticality tasks. Under normal conditions, these lower criticality tasks can resume using the cycles not required by the higher criticality ones.

Additional complexities arising from multicore systems include a new memory hierarchy with shared caches, and shared memory buses and core interconnects, as well as shared memory. These complexities have an impact on the execution latency, particularly in the on-chip and off-chip memory bandwidth that is achievable. The SEI has been exploring different approaches to address these issues [Andersson 2012a].

Similarly, power consumption is now of significant interest in small defense vehicles (such as UAVs) that have limited power capacities, and, for instance, running on batteries. For this case, the SEI has developed an optimal algorithm for the selection of the frequency at which cores in a multicore processor should be run [Moreno 2012].

Current SEI research is focused on scheduling schemes for parallelized tasks that need to be executed in multiple cores simultaneously in order to meet their deadlines [Andersson 2012b]. This imposes not only a challenge to decide which parts of a task need to be executed in which core, but also how to analyze the use of shared memory that can cause delays in one core when a task in another core accesses the same region of memory.

### 2.7.3 The Consequence: Effective Use of Multicore Processors

The scheduling techniques being developed at the SEI enable practitioners to verify real-time systems using multicore processors. This is of critical significance, since the current practice has been to either avoid the use of multicore processors or disable all processors except one so that old techniques work.

The SEI work has triggered the interest of both NASA and Lockheed Martin Aeronautics, and the SEI is investigating potential applications in their settings. SEI papers on mixed-criticality scheduling are among the most-cited papers in the literature.

### 2.7.4 The SEI Contribution

This research is being conducted in collaboration with Carnegie Mellon research faculty. All members of the team have made previous contributions to the theory and practice of scheduling for real-time systems. Since all results are those of the full team, it is not possible to isolate the specific contributions made by the SEI.

SEI papers on mixed-criticality scheduling are among the most-cited papers in the literature.

### 2.7.5 References

[Anderson 2006] Anderson, J.; Calandrino, J. M.; & Devi, U. C. "Real-Time Scheduling on Multicore Platforms," 179-190. *Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium*, San Jose, CA, April 4-7, 2006. IEEE, 2006.

[Andersson 2001] Andersson, B.; Baruah, S.; & Jonsson, J. "Static-Priority Scheduling on Multiprocessors," 193-202. *Proceedings of the IEEE Real-Time Systems Symposium*, London, December 3-6, 2001. IEEE, 2001.

[Andersson 2003] Andersson, B.; Abdelzaher, T.; & Jonsson, J. "Global Priority-Driven Aperiodic Scheduling on Multiprocessors." International Parallel and Distributed Processing Symposium, Nice, France, April 22-26, 2003. IEEE, 2003.

[Andersson 2012a] Andersson, Bjorn; Chaki, Sagar; de Niz, Dionisio; Daugherty, Brian; Kegley, Russell; & White, Jules. "Non-Preemptive Scheduling with History-Dependent Execution Time," 363-372. *Euromicro Technical Committee on Real-Time Systems (ECRTS).* Pisa, Italy, July 11-13, 2012. IEEE, 2012.

[Andersson 2012b] Andersson, Bjorn & de Niz, Dionisio. "Analyzing Global-EDF for Multiprocessor Scheduling of Parallel Tasks." *16th International Conference on Principles of Distributed Systems (OPODIS)*. Rome, Italy, December 17-20, 2012. Springer, 2012.

[de Niz 2009] de Niz, Dionisio; Lakshmanan, Karthik; & Rajkumar, Raj. "On the Scheduling of Mixed-Criticality Real-Time Tasksets." *IEEE Real-Time Systems Symposium (RTSS)*. Washington, D.C., December 1-4, 2009. IEEE, 2009.

[Dhall 1978] Dhall, K. & Liu, C. L. "On a Real-Time Scheduling Problem." *Operations Research 26*, 1: 127-140.

[Lakshmanan 2009] Lakshmanan, Karthik; de Niz, Dionisio; & Rajkumar, Raj. "Coordinated Task Scheduling, Allocation, and Synchronization in Multiprocessors." *IEEE Real-Time Systems Symposium.* Washington, D.C., December 1-4, 2009. IEEE, 2009.

[Lakshmanan 2010] Lakshmanan, Karthik; de Niz, Dionisio; & Rajkumar, Raj. "Resource Allocation in Distributed Mixed-Criticality Cyber-Physical Systems." 169-178. *30th International Conference on Distributed Computing Systems (ICDCS)*, Genoa, Italy, June 21-25, 2010. IEEE, 2010.

[Lakshmanan 2011] Lakshmanan, Karthik; de Niz, Dionisio; & Ragunathan (Raj) Rajkumar. "Mixed-Criticality Task Synchronization in Zero-Slack Scheduling," 47-56. *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. Chicago, IL, April 11-14, 2011.

[Moreno 2012] Moreno, Gabriel & de Niz, Dionisio. "An Optimal Real-Time Voltage and Frequency Scaling for Uniform Multiprocessors," 21-30. *IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. Seoul, Korea, August 20-22, 2012. IEEE, 2012.

[Srinivasan 2001] Srinivasan, A. & Anderson, J. "Optimal Rate-Based Scheduling on Multiprocessors," 189-198. *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*. Heraklion, Crete, Greece, July 6-8, 2001. ACM, 2001.

## 2.8 Integrated Methods for Predictive Analytic Composition and Tradeoff

### 2.8.1 The Challenge: Effective Real-Time Performance in Dynamic Environments

DoD missions often require systems that provide effective real-time performance in dynamic environments under constrained conditions. This performance requires designing systems to get maximum utility from limited resources by performing both appropriate tradeoffs at design time and appropriate adaptation at runtime.

In 2000, DARPA was conducting a research program designed to develop and transition critical technology that would enable mission-critical systems to meet high-assurance, high-dependability, high-adaptability DoD requirements. This was the Dynamic Assembly for System Adaptability, Dependability, and Assurance (DASADA) Program [Mandak 2001]. The integrated methods for predictive analytic composition and tradeoff (IMPACT) project was funded to contribute to this program.

### 2.8.2 A Solution: Development of Analytic Methods

The IMPACT project was a joint effort between the CMU School of Computer Science, the SEI, and Lockheed Martin Aeronautics (LM-Aero). The goal was the development of analytic methods to support the correct temporal composition of systems. The methodology focused on techniques to construct systems having predictable timing performance and composed of pre-analyzed components. The theory developed by this project formed an engineering basis for the design of real-time systems. The targeted capability was the rapid assembly of complex, high-assurance DoD systems operating in dynamic environments and having critical timing requirements and computing resource constraints. This capability could also be used to support both engineering tradeoff analyses across multiple performance dimensions at system design time, and runtime adaptation to mission, environment, and computing resource changes.

Lockheed Martin developed a model avionics problem applicable to the F-22 and F-35 Joint Strike Fighters (JSF). Researching solutions to this problem was one direction of the IMPACT work.

The team developed the temporal analytic composition theory (TACT), researching several technologies and integrating them into a powerful composition methodology. The component technologies were generalized rate monotonic analysis (GRMA), real-time queueing theory (RTQT), quality-of-service resource allocation methodology (Q-RAM), and hierarchical scheduling.

*Generalized rate monotonic analysis*, as explained in another subsection, is a real-time scheduling methodology designed to handle tasks with hard-deadline periodic tasks and soft-deadline aperiodic tasks. While GRMA offers a flexible resource management architecture, its application leads to very conservative resource requirements. IMPACT technology and tools were used to improve on this issue, offering high levels of resource utilization while preserving the system requirements. One such tool was TimeWiz.

*Quality-of-service resource allocation methodology* maximizes total system utility with limited resources. *Visual Q-RAM is* an engineering tool that supports design-time prediction and tradeoff

analysis as well as model tuning for runtime deployment of Q-RAM models. Guidance was developed for model tuning of the Q-RAM. Model tuning involves specifying a consistent set of Q-RAM model parameters, including the utility values for quality dimensions and task weights. A notation was also developed to support the expression of cross-task quality-level constraints to achieve predictable degradation behavior [Rajkumar 1997]. The Visual Q-RAM software tool was enhanced to support use-scenario walkthroughs and development of models from predefined libraries of task types. Use of this technology in DoD contexts was demonstrated by developing a set of examples; these included helicopter pilot mission support, phased array radar bandwidth allocation, and radar target-tracking algorithm selection.

*Real-time queueing theory* provides accurate timing behavior predictions of real-time systems having stochastic workloads. The theory can be used to assess the ability of a system to meet the timing requirements under heavy traffic conditions. It complements scheduling theories such as generalized rate monotonic scheduling. Visual RTQT is a tool that demonstrates the practicality of using RTQT. Important progress was made in the development of RTQT and its application to avionics and communication systems [Lehoczky 1996]. RTQT is a significant innovation in the design and scheduling of real-time systems in that it is capable of making exact predictions of a system's ability to meet the timing requirements of real-time tasks where task arrivals and computation requirements are stochastic. It extends methodologies such as GRMA to a greatly broadened framework. The RTQT project innovations include development of an RTQT-based analysis of the temporal behavior of the F-22 avionics challenge problem and an RTQT analysis for feed-forward queuing networks and for acyclic networks, leading to an analysis tool for the determination of end-to-end schedulability requirements.

*Hierarchical scheduling* is a method based on GRMA that enables a single schedulable physical resource, such as a processor, to be partitioned into multiple isolated virtual resources. Different algorithms and analysis techniques can be used in each of the virtual resources, and changes in the temporal properties within one virtual resource do not impact the temporal behavior within other virtual resources [Saewong 2002].

**The View from Others**

*This group [CMU IMPACT] demonstrated all proposed objectives from the DASADA literature. This group works closely with Lockheed Martin on real time scheduling and context testing on the F-16 avionics platforms. CMU is doing breadboard testing and creating prototypes for a new advanced avionics suite proposed for future aircraft development. An evaluation of this system indicated this group is ready to move on to the next phase of the DASADA program. (pp. 63-64)*

*Out of the 19 projects, there is only a handful that should be considered for future funding based upon their level of effort over the past several months, as well as their level of technology maturity to be able in the next year to actually provide a component to insert into the DASADA Dynamic Assembly Toolkit. [One of those projects is] CMU's Integrated Methods for Predictive Analytic Composition and Tradeoff (IMPACT). (pp. 76-77)*

– Wayne S. Mandak and Charles A. Stowell, [Mandak 2001]

The benefits of the IMPACT methodology and associated tools to DoD systems were made apparent in two demonstrations: a phased array radar scheduling demonstration and an embedded avionics simulation designed to support the F-22 and F-35 JSF programs.

The first demonstration (at the DASADA Demonstration and Exposition in July 2002) showed the runtime use of Q-RAM. A visual demonstration represented a carrier under attack from a set of enemy locations. With a variety of threats attacking the ship, the phased-array radar had to be scheduled in such a way that the ship could destroy those threats. The radar, treated as a non-preemptible resource, could be allocated to tracking targets in varying amounts to achieve different levels of service quality or tracking error [Hansen 2004]. The mission-critical embedded system demonstration illustrated the use of RTQT to assess the ability of a system to meet the timing requirements of stochastic task sets (such as mission-awareness applications) under heavy workload conditions. When RTQT indicates that latency requirements are not met, it provides an engineering basis for modifying the workload to achieve the desired latency within a specified degree of certainty.

### 2.8.3 The Consequence: Bringing an Analytic Basis to Engineering Dynamic Systems

Work on the F-22 embedded avionics simulation provided new and valuable temporal and performance analysis information to F-22 designers that was not previously available and that could be used in the final refinement of the F-22 weapon system. Further, it provided insight into several design improvements for legacy systems. System utilization, resource efficiency, and maintainability could be significantly enhanced using DASADA-developed design practices. Progress made under DASADA allowed LM-Aero and CMU to propose a large-scale DASADA II experiment centered on upgrading the F-22 mission computer temporal architecture. Results and insights from this experiment aimed to reduce both new development and application rehost costs through the analytic composition methodology developed under DASADA I. Additional benefits included enhanced adaptability to dynamic mission demands, along with increased system reconfiguration options.

In addition, IMPACT technology was applied in the context of a rotorcraft through collaborations with U.S. Army Aviation and Missile Command (AMCOM). Investigations into integration of Q-RAM with hard real-time scheduling in avionics applications under DASADA led to interest in this technology by the U.S. Army AMCOM. This led to the invitation to join a team to propose a large-scale DASADA II experiment centered on application of DASADA technologies to the Sikorsky Black Hawk helicopter.

### 2.8.4 The SEI Contribution

This research was conducted in collaboration with the Carnegie Mellon faculty and Lockheed Martin Aeronautics Co. All members of the team made contributions to the IMPACT project. Since all results are those of the full team, it is not possible to isolate the specific contributions made by the SEI.

### 2.8.5 References

[Hansen 2004] Hansen, J.; Ghosh, S., Rajkumar, R.; and & Lehoczky, J. “Resource Management of Highly Configurable Tasks,” 116. *Proceedings of the 18th International Parallel and Distributed Processing Symposium*, April 26-30 2004. IEEE, 2004. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1303070&tag=1

[Lehoczky 1996] Lehoczky, John P. “Real-Time Queueing Theory,” 186. *Proceedings of the IEEE Real-Time Systems Symposium.* Washington, DC, December 4-6, 1996. IEEE, 1996. dl.acm.org/citation.cfm?id=828944

[Mandak 2001] Mandak, Wayne S. & Stowell, Charles. *A Dynamic Assembly for System Adaptability, Dependability and Assurance (DASADA) Project Analysis.* Naval Postgraduate School, 2001. http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA393486

[Lehoczky 1996] Lehoczky, John P. “Real-Time Queueing Theory.” *Proceedings of the IEEE Real-Time Systems Symposium.* Washington, DC, December 4-6, 1996. IEEE, 1996. dl.acm.org/citation.cfm?id=828944

[Rajkumar 1997] Rajkumar, R.; Lee, C., & Lehoczky, J., & Siewiorek, D. “A Resource Allocation Model for QoS Management,” 298-307. *Proceedings of the IEEE Real-Time Systems Symposium*. San Francisco, CA, December 3-5, 1997. IEEE, 1997. http://www.cs.cmu.edu/afs/cs/project/rtmach/public/papers/qos.ps

[Saewong 2002] Saewong, S.; Rajkumar, R.; Lehoczky, J. P.; & Klein, M. H. “Analysis of Hierarchical Fixed-Priority Scheduling,” 173-181. *Proceedings of Euromicro Conference on Real-Time Systems.* Vienna, Austria, June 19-21, 2002. IEEE, 2002. http://www.computer.org/csdl/proceedings/ecrts/2002/1665/00/16650173-abs.html

## 2.9 Architecting Software-Reliant, Safety-Critical Systems with SAE AADL

### 2.9.1 The Challenge: Reducing Faults in Safety-Critical Defense Systems

Safety- and mission-critical systems, such as aircraft, motor vehicles, and communication systems, have become increasingly software reliant. The cost of developing such systems has increased exponentially under the current practice of "build then test" and has become unaffordable—reaching 10 billion dollars for the next-generation aircraft, with software comprising 70 percent or more of the total system cost [Redman 2010]. The results are major delays in system delivery and unexpected system failures during operation. A major cost driver is the exponential growth in software size and interaction complexity. This growth is due to the increasing role of software as the integrator of system functionality and the use of a shared networked computer hardware infrastructure. Studies show that for safety-critical software systems, 70 percent of faults are introduced during requirements specification and architecture design; and 80 percent are currently not caught until integration/acceptance testing and actual operation, with rework cost factors of 110-1000x [AVSI 2010]. Many of the root causes are related to mismatched assumptions in the interaction between the software, the hardware, and the physical system [Feiler 2009]. Studies of this problem have recommended a paradigm shift toward an architecture-centric, model-based practice of end-to-end assurance evidence through predictive analysis and formal verification to complement testing [NRC 2007].

### 2.9.2 A Solution: SAE Architecture Analysis & Design Language (AADL)

In the 1990s, DARPA-funded research in software architecture fostered the creation of a number of architecture description languages (ADLs), one of them being MetaH, which was specifically designed at the Honeywell Technology Center for embedded software systems and which supported RMA [Vestal 1993]. Its successful use on a missile guidance system at the U.S. Army Aviation and Missile Research Development and Engineering Center (AMRDEC) Software Engineering Directorate (SED) and several other pilot projects led AMRDEC SED, in 1999, to kick off and chair a standardization effort through the SAE AS-2C Architecture Description Language Committee in the Avionics Systems Division of SAE International. Under the technical leadership of the SEI, the AADL standard was approved by 23 voting member organizations and published in November 2004; it was revised in January 2009 based on feedback from the user community [SAE AADL 2009]. In June 2006, a set of Annex standards was published to support various forms of hazard, reliability, and fault-impact analysis. The Annex included the AADL Meta model and XMI interchange format, and the error model extension to AADL. In January 2011, a second set of Annex standards was published, consisting of a Behavior Annex, a Data Modeling Annex, and an ARINC653. With the release of the standard, the SEI provided an Eclipse-based open source implementation of a tool environment for AADL called OSATE to encourage pilot projects.

SAE AADL was specifically designed to support modeling and analysis of large-scale embedded software system architectures in terms of an application runtime architecture bound to a computer platform architecture and interacting with a physical system in which it is embedded. The architecture is expressed through concepts with well-defined semantics, such as periodic and aperiodic tasks with sampled and queued communication operating as a partitioned system on synchronous

or asynchronous networked computer hardware. The annotated architecture model supports analysis of functional and nonfunctional properties, including schedulability, safety, and reliability, as well as code generation of runtime executives integrated with application components. Derivation of different analytical models from the annotated architecture model assures consistency of analysis results. The ability to model and analyze the architecture early and throughout the development at increasing levels of fidelity leads to an incremental approach of end-to-end system validation and verification.

### 2.9.3 The Consequence: Architecture-Centric Engineering Beyond Documentation

Several industry initiatives initiated pilot projects as soon as the AADL standard was published. The first industrial initiative using AADL as a core technology was the ASSERT project, led by the European Space Agency in cooperation with 29 partners, from 2004-2007. The initiative developed a tool chain for the model-based analysis and auto-generation of satellite systems from these reference architecture models and applied it to two families of satellite systems.

In 2005, a five-year industry initiative of 28 partners, led by Airbus and called TOPCASED, developed an industrial open source tool infrastructure for model-based engineering of embedded systems, with OSATE as part of the tool suite (http://www.topcased.org/). In 2006, the three-year ITEA SPICES initiative of 15 research and industrial partners began to develop a model-based engineering method that integrates modeling in CORBA Component Model (CCM) and AADL for analysis and auto-generation into SystemC.

From 2008-2011, the COMPASS Project (http://compass.informatik.rwth-aachen.de/), an international research project funded by the European Space Agency, developed a theoretical and technological basis and approach for the system-software co-engineering. This co-engineering approach focused on a coherent set of specification and analysis techniques evaluating system-level correctness, safety, dependability, and performability of on-board computer-based aerospace systems. These techniques have significantly improved the reliability of modern and future space missions.

Since 2008, under the umbrella of the Aerospace Vehicle Systems Institute (AVSI)—a consortium of aerospace companies, including Boeing, Lockheed Martin, Airbus, Embraer, and a number of suppliers, including BAE Systems, Rockwell Collins, Honeywell, GE Aviation, as well as the FAA, NASA, and the DoD—started the multi-phase System Architecture Virtual Integration (SAVI) initiative to establish a architecture-centric, model-based “integrate then build” practice throughout the lifecycle. SAVI uses a multi-notation model repository approach that assures model consistency and interchange based on industry standards without forcing participants into the same tool set. For the proof-of-concept phase, AADL and OSATE were chosen as key technologies for a case study to (1) analyze multiple quality attribute dimensions at several levels of fidelity on a multi-tier aircraft model, and to (2) illustrate the ability to support integrator/supplier interactions through architecture model interchange via a model repository [Redman 2010]. SAVI is in the process of performing shadow projects within member companies and establishing buy-in from commercial tool venders into the SAVI approach.

### 2.9.4 The SEI Contribution

As technical lead of the SAE AADL standard, the SEI integrated several research technologies into the AADL standard, making it an extensible, semantically well-defined, and consistent standard suite. Through strong participation and feedback of potential users of AADL from the avionics and space industries, features have been included in the AADL standard to accommodate application to large-scale systems. Through the creation of OSATE, the SEI has fostered pilot applications of AADL in a range of industrial pilot projects. It also has fostered the use of AADL and the OSATE toolset as a technology transition platform, as evidenced by the integration of a number of formal analytical frameworks with AADL (https://wiki.sei.cmu.edu/aadl).

The SEI has created presentation and training materials for architecture-centric, model-based engineering with AADL, applied AADL in a number of customer pilot projects, developed the Virtual Upgrade Validation method to investigate known root cause problem areas in embedded software systems, and used AADL in combination with other SEI architecture-centric methods.

The SEI continues to work with the aerospace industry on the SAVI initiative and the transition of this model-based, architecture-centric practice into military programs.

### 2.9.5 References

[AVSI Feiler 2010] Feiler P.; Wrage L.; & Hansson J. “Toward Model-Based Embedded System Validation through Virtual Integration.” DoD Data Analysis Center for Software (DACS*). Journal of Software Technology* (January 2010).

[Feiler 2009] Feiler, Peter H. “Challenges in Validating Safety-Critical Embedded Systems.” *Proceedings of SAE International AeroTech Congress*, Seattle, WA, November 10-12, 2009.

[NRC Jackson 2007] Jackson, Daniel, ed. *Software for Dependable Systems: Sufficient Evidence?* Committee on Certifiably Dependable Software Systems, National Research Council. National Academic Press, 2007 (ISBN: 0-309-10857-8).

[Redman 2010] Redman, David; Ward, Donald; Chilenski, John; & Pollari, Greg. “Virtual Integration for Improved System Design.” *Proceedings of The First Analytic Virtual Integration of Cyber-Physical Systems Workshop* in conjunction with the Real-Time Systems Symposium (RTSS 2010), San Diego, CA, November 30-December 3, 2010.

[SAE AADL 2009] Society of Automotive Engineers (SAE) Avionics Systems Division (ASD) AS-2C Subcommittee. *Avionics Architecture Description Language Standard.* SAE Documents AS 5506A in January 2009, AS 5506/1 in June 2006, and AS 5506/2 in January 2011. http://www.sae.org.

[SEI News 2009] Software Engineering Institute. “Creating a Framework for Reliability Validation” (SEI News story). November 24, 2009. http://www.sei.cmu.edu/newsitems/amrdec_roadmap.cfm

[Vestal 1993] Vestal, S. & Binns, P. “Scheduling and Communication in MetaH,” 194-200. *Proceedings of the Real-Time Systems Symposium.* RSTS, Durham, NC, December 1-3, 1993. IEEE, 1993.

# 3 Education and Training

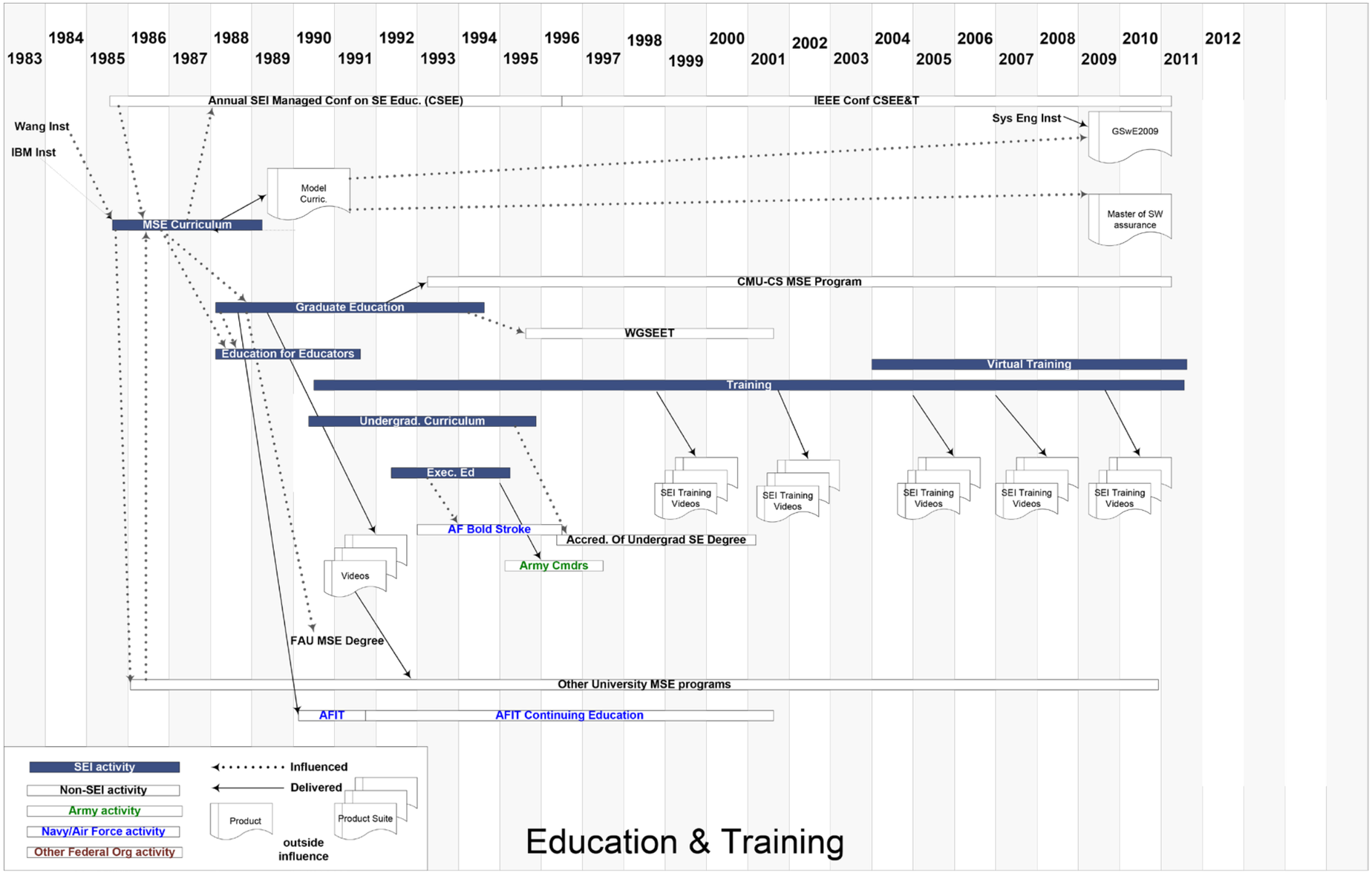


*Figure 4: Education and Training Timeline*

## 3.0 Introduction to Education and Training

A major factor in the ability of the Department of Defense to acquire and maintain software-intensive systems is the availability of properly educated software engineers, not just in the DoD but in the supporting industry. Likewise, a major component of technology transition is the availability of training for practicing engineers and the availability of training materials for use by third-party training organizations.

The DoD recognized these needs and included the following in the Software Engineering Institute's initial contract: "The SEI shall develop and conduct courses and seminars with respect to the evolving state of the art and practice in software engineering for mission-critical computer systems as well as the results of its activities in technology transition. It shall also influence software engineering curricula development throughout the education community" [DoD 1984].

Part of the motivation for this charge to influence curricula development was the recognition that there was no widely accepted curriculum for preparing students for a career in software engineering. There were only two university programs offering a Master of Software Engineering (MSE) degree and a few scattered university software engineering courses, but most universities did not even offer such courses and few faculty were prepared to teach them. Several companies, such as IBM with its Software Engineering Institute, conducted their own training programs; but that training was specifically for internal use.

A strong factor for the government's selection of Carnegie Mellon University for the operation of the SEI was its established capability and reputation for engineering and computer science education. There was, and still is, recognition in the DoD that it is not possible to properly educate software engineers and improve the state of the practice without a strong education and technology transition capability.

As a federally funded research and development center and as an objective broker of information, the SEI develops methodologies and training that are neither vendor specific nor vendor biased. SEI training is driven by the needs of the government and supporting industry, independent of commercial needs or profit motives.

## 3.1 Academic Curricula

Charged with the mission to influence software engineering curriculum development throughout the education community, the SEI recruited a software educator to lead the effort. Recognizing that the effort would be successful only if it involved a broad segment of the academic community, the SEI initiated a series of workshops [Gibbs 1989], inviting educators and practicing engineers to develop "curriculum modules." This led to the model curriculum for a Master of Software Engineering (MSE) degree [Gibbs 1990], which is the basis for MSE programs at many universities. It is also the model for other curricula developed much later, such as the Graduate Software Engineering Curriculum (GswE2009) developed by the Systems Engineering Research Center (SERC) in 2009.

Similar to its work in other areas, the SEI engaged the academic community in creating the materials. The SEI has leveraged and amplified technology transition with government and industry by

making materials available to allow other organizations to teach material it has developed. For instance, the Air Force Institute for Technology (AFIT) sent six officers from its faculty to the SEI as resident affiliates to adapt the SEI MSE program. AFIT then offered software engineering education as part of its continuing education program.

As the number of MSE programs began to rise, some universities incorporated material from the curriculum modules in undergraduate courses. Although initially software engineering was thought too advanced for an undergraduate curriculum, as the MSE programs matured, demand for an undergraduate curriculum began to build. The SEI was asked to lead development of an undergraduate curriculum in software engineering. The SEI engaged the stakeholders through the Association for Computing (ACM) and the Computer Society of the Institute of Electrical and Electronics Engineers (IEEE-CS) in joint efforts to develop curriculum guidelines for undergraduate programs in computing: computer engineering, computer science, information systems, information technology, and software engineering. The Accreditation Board for Engineering and Technology (ABET) has since established Software Engineering Program Criteria and, as of 2012, accredited 27 programs.

The Working Group on Software Engineering Education and Training (WGSEET) was formed in 1995. The WGSEET was an ad hoc group of approximately 80 international professionals from academia, industry, and government, led by the SEI's former education director. The WGSEET engaged in an extensive set of activities in support of undergraduate education. It developed guidance and support for the development of undergraduate computing curricula to support the education of software engineering professionals [Bagert 1999]; formulated ideas and issues related to the design and implementation of a curriculum for the introductory part of a BS degree in software engineering [Hilburn 2003]; interacted with the ACM and IEEE-CS; provided support for the development of an undergraduate software engineering curriculum model; and participated in numerous workshops and panels at the Conference on Software Engineering Education and Training (CSEET).

Based on the SEI's contribution to curricula development and its expertise in software security, the U.S. Department of Homeland Security (DHS) sponsored the SEI to build a model curriculum for software assurance education and define strategies to implement it. Following an approach similar to previous curriculum development efforts, the SEI led development of a Software Assurance Curriculum for Colleges and Universities intended for graduate education leading to a master's degree [Mead 2010a]. In 2011 the curriculum was recognized by the IEEE and the ACM professional societies as a model curriculum for a master's degree program in software assurance, a significant achievement. The curriculum work was subsequently extended to address the educational needs at the bachelor and associate degree levels [Mead 2010b] drawing on the body of knowledge defined in the master's curriculum. As of 2012, software assurance courses were being offered at several universities, including the U.S. Air Force Academy. In 2013, the SEI added online, on-demand software assurance training for executives.

The National Guard asked the SEI to develop a curriculum for survivability and information assurance education for systems administrators appropriate for the community college level. The resulting four-course curriculum and capstone project enabled the National Guard to encourage local community colleges to offer this program for National Guard systems administrators and also

to the colleges' broader student population. As a result, the National Guard gained highly qualified systems administrators who could address the survivability and information assurance of their systems, and additional well-qualified system administrators entered the workforce. The materials were made available online and had widespread influence. For example, as of 2014 access to the faculty version of the curriculum was granted to 369 qualified faculty members representing 239 colleges and universities in 43 U.S. states, the District of Columbia, and one Canadian province.

### 3.1.1 Curricula Transition

The SEI approach to curriculum development by engaging the academic community provided a natural mechanism for transition. In addition, the SEI engaged other established mechanisms for transition. Faculty development workshops were conducted in conjunction with a Conference on Software Engineering Education. In a tutorial format, the SEI provided materials and preparation to faculty members planning to teach software engineering and continued refinement of the model curriculum. Sponsorship of the conference was eventually transferred to the IEEE, and continues today as the IEEE Conference on Software Engineering Education and Training. The CSEE led to the first electronic newsletter for software engineering educators, FASE (Forum for Advancing Software Engineering Education). The Working Group on Software Engineering Education and Training was also an outgrowth of the CSEET. The working group met twice yearly between 1995 and 2000, producing a number of education reports and other artifacts.

Another outgrowth of the curriculum project was the development of curriculum modules and educational materials, which helped to transition the MSE curriculum and support faculty members who wished to offer software engineering course offerings. This mentoring relationship helped to transition the MSE curriculum and establish many of the software engineering degree programs that exist today. More than 50 graduate MSE programs have been created; they produced more than 700 graduates in 2004 alone [Ardis 2005].

Feedback from the university community suggested that although the model curriculum and supporting materials were helpful, individual colleges and universities would be more motivated to begin their own programs if a university of CMU's stature were to do so as well. The SEI responded by teaming with the CMU School of Computer Science (SCS) to offer an MSE housed at the SEI, with the degree being granted by SCS. A unique characteristic of the program was (and is today) a design studio modeled after the design studios used in architecture. Under the guidance of a senior faculty member, students undertake realistic design projects offered by industry and government organizations. The CMU Master of Software Engineering Program [Gibbs 1990] continued as a joint program for several years, graduating professional engineers who were highly recruited by industry. The MSE program was eventually transferred to SCS where it continues today. SEI staff members continue to be involved, serving as studio mentors and occasionally teaching courses.

When the SEI and SCS joined to create the MSE program, the SEI recognized the opportunity to increase its support to other universities by videotaping the CMU courses and offering them to other colleges and universities. The SEI recruited the retiring manager of the IBM Software Engineering Institute. Recognizing that it was neither practical for SEI people to travel to other universities nor for university faculty from other universities to spend significant time at the SEI, the SEI embarked on the construction of a video studio. Courses were taped live with CMU students in the classroom and provided to other universities. Some universities showed the videos as a

course offering, while at other universities, faculty reviewed the videos and then offered similar, but tailored, lectures to their students.

A series of train-the-trainer continuing education courses was developed in conjunction with the video studio. These courses mirrored the academic offerings on topics such as software project management and software requirements engineering, among others. Instructors who attended these courses were trained in the complete set of materials, and received a set of videotapes, slides, lecture notes, and other supporting materials for delivery of the continuing education offerings.

The SEI also recognized that successful development of software engineering in the academic community required definitive books from which faculty could become familiar with evolving thought in the field. The SEI, therefore, teamed with Addison-Wesley to publish the SEI Series in Software Engineering. Today the series comprises more than 30 volumes on a wide range of software engineering topics. Many of the books in the series are authored by members of the SEI staff.

The formal education component was transitioned over time to the academic infrastructure. Colleges and universities were offering degrees based on accredited curricula. The workshops and conferences at which professors could update their knowledge were transitioned to the IEEE [Mead 2009]. As a consequence, although the SEI remains active in software engineering education, it is no longer a major focus for the SEI. It is, however, a major success of its original charter that software engineering and related programs are so vibrant in the academic infrastructure.

### 3.1.2 Professional Education and Training

With the success of the academic programs, a demand for executive education began to surface. Defense industry and DoD executives whose expertise was in areas other than software began to request executive education that would enable them to understand the issues associated with managing software-intensive systems. The SEI recruited a retired industry vice president and a retired flag officer to develop an executive education program. The executive education offerings were extremely popular with senior executives, and the Air Force adapted the material for a program called Bold Stroke for Officers at the rank of 0-6 and above. Recognizing the role software played in Desert Storm, the Army asked the SEI to offer a version of its executive education program to Army flag officers and Senior Executive Service (SES) civilians; the program is offered annually to the Army senior leadership.

The SEI has provided a broad array of professional training. In addition to the courses offered by SEI staff, the SEI has established a partner program that includes teaching SEI professional education courses. Approximately 400 partner organizations are authorized to teach SEI courses. In 2010, partners taught 15,000 students, while the SEI taught 4,000 students—highlighting the value of leveraging SEI education and training through collaborations and partnerships.

The SEI also offers executive education programs for commercial organizations. The SEI developed and hosted “Technovation,” a weeklong executive training lab specifically designed to support General Electric’s (GE’s) Experienced Information Management Program (EIMP). Technovation participants represented a range of GE divisions and information technology roles. The executive training was delivered by experts from the SEI and guest lecturers from Carnegie

Mellon's faculty in areas such as robotics, interdisciplinary collaboration, team development, software architecture and product line development, service-oriented architecture, data mining, and cybersecurity [SEI 2012].

### 3.1.3 Evolution of Instructional Delivery Based on Technology Advancements

The delivery of SEI education and training has evolved, taking advantage of advances in technology as they occur. The SEI can now deliver training to warfighters as well as others across the globe 24/7, and students can learn at their own pace and study at home, work, or on the road.

This evolution of training delivery methods has enabled the SEI and the government to provide timely training at an affordable cost. The SEI video studio was used to record lectures by leaders in the software engineering community to provide insight into important developments, resulting in a standalone video series. The SEI has added to traditional instructor-led classroom training by taking advantage of synchronous distance training such as broadcasting virtual "live" online training, and, later, asynchronous web-based, self-paced e-learning. Recently, the SEI has been producing podcasts and webinars to keep pace with the latest evolution from e-learning to m-learning—on-the-go mobile learning.

The SEI found that its customers' unique distance training needs required new technology for cyber workforce development. The CERT Virtual Training Environment (VTE) was developed to amplify the security training and best practices delivered through classroom training. Because of its rich media instruction and hands-on training labs, VTE allowed users to access high-quality training materials in information security, computer forensics, and incident response anywhere in the world with only a web browser and an internet connection.

The CERT Exercise Network (XNET) solved the problem of preparing staff to train under realistic conditions, using scenarios that can be difficult and expensive to create and administer. XNET allowed organizations to create customized, realistic, interactive simulations on an isolated network. Through a web-based interface, participants across multiple locations could work together to analyze and respond to the latest threats. Instructors could easily monitor, control, and evaluate participants' activities to identify problem areas. XNET was used in multiple cyber defense exercises conducted by the U.S. government.

Features of VTE and XNET are included in STEPfwd, the most recent advance in technology for workforce development. Like its predecessor, STEPfwd is being used in military cyber defense exercises.

### 3.1.4 References

[Ardis 2005] Ardis, Mark. "An Incomplete History of Master of Software Engineering Programs in the United States." Presentation at the 15th Reunion of CMU MSE Program, July 2005. http://personal.stevens.edu/~mardis/papers/MSEHistory.pdf

[Bagert 1999] Bagert, D.; Hilburn, T.; Hislop, G.; Lutz, M.; & McCracken, M. "Guidance for the Development of Software Engineering Education Programs." *The Journal of Systems and Software 49* (1999): 163-169.

[DoD 1984] Statement of Work for Implementation and Initial Operations of the DoD Software Engineering Institute, Section J Attachment 1 F19628-84-R-0070, May 30, 1984. Not publicly available.

[Gibbs 1989] Gibbs, N. E. “The SEI Education Program: The Challenge Of Teaching Future Software Engineers.” *Communications of the ACM 32,* 5 (May 1989): 594–605.

[Gibbs 1990] Gibbs, Norman E.; Ardis, Mark A.; Habermann, A. Nico; & Tomayko, James E. “The Carnegie Mellon University Master of Software Engineering Degree Program.” 152-154. *Proceedings of the Software Engineering Education Conference*, Pittsburgh, PA, April 2-3, 1990. Published as Springer Lecture Notes in Computer Science 423, 1990.

[Hilburn 2003] Hilburn, T.; Duley, D.; Hislop, G.; & Sobel, A. “Engineering an Introductory Software Engineering Curriculum,” 99–106. *Proceedings of the Sixteenth Conference on Software Engineering Education and Training*. CSEET, Madrid, Spain, March 20-22, 2003. IEEE, 2003.

[Mead 2009] Mead, N. R. “Software Engineering Education: How Far We’ve Come and How Far We Have to Go.” *Journal of Systems and Software* (2009), doi:10.1016/j.jss2008.12.038.

[Mead 2010a] Mead, Nancy; Allen, Julia; Ardis, Mark; Hilburn, Thomas; Kornecki, Andrew; Linger, Richard; & McDonald, James. *Software Assurance Curriculum Project Volume I: Master of Software Assurance Reference Curriculum* (CMU/SEI-2010-TR-005). Software Engineering Institute, Carnegie Mellon University, 2010. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9415

[Mead 2010b] Mead, Nancy; Hilburn, Thomas; & Linger, Richard. *Software Assurance Curriculum Project Volume II: Undergraduate Course Outlines* (CMU/SEI-2010-TR-019). Software Engineering Institute, Carnegie Mellon University, 2010. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9543

[SEI 2012] “SEI Bolsters GE Professional Development Curriculum with ‘Technovation’ Executive Training Lab.” (Press Release) Software Engineering Institute, Carnegie Mellon University, 2012. http://www.sei.cmu.edu/newsitems/technovation-2011.cfm

## 3.2 Model Curriculum for Master of Software Engineering Degree

### 3.2.1 The Challenge: The Need for a Standard Software Engineering Curriculum

One of the outcomes of the NATO workshop [Naur 1969] that coined the term "software engineering" was an interest in developing new educational programs, especially for working professionals. However, new programs were slow to appear. One problem was the lack of a standard curriculum.

In the late 1970s, the IEEE Computer Society formed a Subcommittee on Model Curricula in Software Engineering [Fairley 1978]. This effort was influenced by earlier work on software engineering education that focused on the skills needed by practicing software engineers [Freeman 1976]. Unfortunately, the IEEE Computer Society never officially endorsed the proposed curriculum from this committee. However, committee members helped start Master of Software Engineering (MSE) programs at Seattle University and Wang Institute of Graduate Studies using the material from the committee's report [Ardis 1987].

### 3.2.2 A Solution: Creation of the SEI Master of Software Engineering Curriculum Guidelines

In 1987, the SEI published some guidelines for graduate programs [Ford 1987] that specified important topics to include in an MSE curriculum. It also provided advice on educational objectives, prerequisites, electives, and needed resources. That report served as a specification for graduate software engineering programs, but it did not propose any specific courses or suggest how they might be taught.

In the winter of 1988, the SEI held a workshop of leading software engineering educators to design a recommended curriculum for an MSE degree. They assumed that such a program would consist of about 10 to 12 courses: six or seven required courses, another three or four electives, with the remainder devoted to project work.

The members of the workshop first estimated the size of each of the 20 topics from the specification, then partitioned them into six core courses:

1. Software Systems Engineering
2. Specification of Software Systems
3. Principles and Applications of Software Design
4. Software Generation and Maintenance
5. Software Verification and Validation
6. Software Project Management

Preliminary descriptions of each of the courses were written during the workshop. After the workshop, a subset of the participants prepared detailed descriptions of the core courses. For each course, they created a catalog description, a statement of educational objectives, an outline of topics, and other supporting material. The student-expected outcomes and required classroom times were also provided for each major topic. All the core courses were independent, with none being a prerequisite for any of the others.

The curriculum guidelines also suggested that about 20-40 percent of an MSE program should consist of electives, and another 30 percent should be project work. Rather than recommend any particular project course design, the guidelines described alternatives used by some of the existing software engineering programs.

Finally, the curriculum guidelines recommended prerequisite knowledge needed by students entering MSE programs, consisting of material in discrete mathematics and computer science topics.

The SEI curriculum recommendations were published at the annual Conference on Software Engineering Education and Training [Ardis 1989], a series started by the SEI that continues today with its own independent steering committee and sponsorship. Some additional material was added to the guidelines in later years, primarily summaries of the courses as they were taught in the Carnegie Mellon MSE program. In 2009, an update to the guidelines [Pyster 2009] was published by an international project. Those guidelines have been adopted by the ACM and the IEEE Computer Society as part of their computing curricula series.

### 3.2.3 The Consequence: New Academic Programs Established

After the SEI guidelines were published, several other schools created their own graduate software engineering programs. In fact, the number of programs nearly doubled in the first three years after the publication of the guidelines. Most of those programs followed the recommended guidelines.

Another outgrowth of the curriculum project was the development of curriculum modules and educational materials that helped to transition the MSE curriculum and support faculty members who wished to offer software engineering courses. This mentoring relationship helped to transition the MSE curriculum and to establish many of the software engineering degree programs that exist today. More than 50 graduate master's programs have been created; they produced more than 700 graduates in 2004 alone [Ardis 2005].

### 3.2.4 The SEI Contribution

The SEI education effort provided needed leadership during the early years of curriculum development in software engineering education. In addition to publishing the first set of curriculum guidelines, the SEI created many early educational resources, such as curriculum modules. These were disseminated at the annual CSEET.

Similar to its work in other areas, the SEI engaged the academic community in creating the materials and leveraged and amplified technology transition with government and industry by making materials available to allow other organizations to teach material it has developed. For instance, the Air Force Institute for Technology (AFIT) sent six officers from its faculty to the SEI as resident affiliates to adapt the SEI MSE program. AFIT then offered software engineering education as part of its continuing education program.

### 3.2.5 References

[Ardis 1987] Ardis, Mark. "The Evolution of Wang Institute's Master of Software Engineering Program." *IEEE Transactions on Software Engineering 13*, 11 (November 1987): 1149-1155.

[Ardis 1989] Ardis, M. & Ford, G. “SEI Report on Graduate Software Engineering Education.” *Proceedings of the 3rd SEI Conference on Software Engineering Education.* CSEE, Pittsburgh, PA, July 18-21, 1989. Published as Springer Lecture Notes in Computer Science 376, 1989.

[Ardis 2005] Ardis, Mark. “An Incomplete History of Master of Software Engineering Programs in the United States.” Presentation at the 15th Reunion of CMU MSE Program, July 2005. http://personal.stevens.edu/~mardis/papers/MSEHistory.pdf

[Fairley 1978] Fairley, R. E. “Toward Model Curricula in Software Engineering,” 77-79. *Proceedings of the 9th SIGCSE Technical Symposium on Computer Science Education.* Published in A*CM SIGCSE Bulletin 10*, 3, (August 1978).

[Ford 1987] Ford, Gary; Gibbs, Norman; & Tomayko, James. *Software Engineering Education: An Interim Report from the Software Engineering Institute* (CMU/SEI-87-TR-008). Software Engineering Institute, Carnegie Mellon University, 1987. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=10253

[Freeman 1976] Freeman, P.; Wasserman, A. I.; & Fairley, R. E. “Essential Elements of Software Engineering Education,” 116-122. *Proceedings of the 2nd International Conference on Software Engineering,* October 1976.

[Naur 1969] Naur P. & Randell, B., eds. *Software Engineering: Report of a Conference Sponsored by the NATO Science Committee.* Garmisch, Germany, October 7-11, 1968. Scientific Affairs Division, NATO, 1969.

[Pyster 2009] Pyster, A., ed. *Graduate Software Engineering 2009 (GSwE2009) Curriculum Guidelines for Graduate Degree Programs in Software Engineering*. Integrated Software & Systems Engineering Curriculum Project, Stevens Institute of Technology, 2009.

## 3.3 Undergraduate Software Engineering Curriculum

### 3.3.1 The Challenge: Lack of Curriculum Guidance for Undergraduate Software Engineering Education

As software engineering became prominent in the 1970s and 1980s, both as a discipline and as a profession, there was little direction or guidance on how to prepare for a career as a software engineer. Through the 1970s, the emphasis in software development was on programming and testing, with little attention to architecture, quality control, process, and the discipline expected in a major engineering activity.

The early SEI education effort focused its work on developing support for a professional master's degree in software engineering [Gibbs 1989]. It soon became clear that the curriculum modules aimed at the master's level could be adapted to support undergraduate-level work.

By the 1990s, the educational community was accepting the idea of software engineering as a separate discipline. In its study and analysis of software engineering's maturity as an engineering discipline, the SEI found that in the U.S. there were approximately 20 universities offering a master's degree in software engineering, but there were no bachelor's degree programs in software engineering [Ford 1996]. (In 1996, Rochester Institute of Technology established the first Bachelor of Science in Software Engineering (BSSE) degree program in the United States.) Undergraduate computer science programs typically contained little or no software engineering material beyond the unit or module development level; there was no meaningful coverage of software requirements, architecture, quality, process, or management topics. In addition, the SEI observed that there was a great variety of educational backgrounds among practicing software engineers, few with formal preparation in software engineering; ABET had not established program criteria for software engineering; and there was no published code of ethics for software engineering. The SEI concluded that initial professional education of software engineers was in the ad hoc stage.

**The View from Others**

*Drexel University has long been a leader in computing education. The University created a master's degree in software engineering in 1997 and followed that effort with a bachelor's degree in software engineering in 2001. Both of these degrees generated considerable discussion and some strong differences of opinion among the faculty. The role of the SEI in establishing the importance of software engineering education was very useful in advancing the effort to create degree programs. In addition, participation in the WGSEET, and availability of all the work products already mentioned provided an excellent vehicle for knowledge sharing to ensure that the Drexel degree programs were in synch with the evolving concepts of software engineering education.*

– Dr. Gregory W. Hislop, Professor, Drexel University

### 3.3.2 A Solution: Development and Dissemination of Curriculum Guidance

The SEI approached the curriculum challenge as it has many others—namely, engage the stakeholders in an effort to meet the challenge. For the past 20 years, the ACM and the IEEE Computer Society have engaged in joint efforts to develop curriculum guidelines for undergraduate programs in computing—computer engineering, computer science, information systems, information technology, and software engineering. Although the 1991 ACM/IEEE-CS computing curriculum

did not list software engineering as a fundamental subject area, the 2001 version identified software engineering as one of the 14 core areas in the computer science body of knowledge [ACM 2001].

The Working Group on Software Engineering Education and Training was formed in 1995 with the mission of improving the state of software engineering education and training. The WGSEET was an ad hoc group of approximately 80 international professionals from academia, industry, and government, which was led by the individual who was the SEI education director from 1991 to 1994 [Bagert 2008]. The WGSEET engaged in an extensive set of activities in support of undergraduate education. It developed guidance and support for the development of undergraduate computing curricula for the education of software engineering professionals [Bagert 1999a]; formulated ideas and issues related to the design and implementation of a curriculum for the introductory part of a B.S. degree in software engineering [Hilburn 2003]; interacted with the ACM and IEEE-CS; provided support for the development of an undergraduate software engineering curriculum model; and participated in numerous workshops and panels at CSEET conferences.

**The View from Others**

*The consequence of the SEI's work in education—model curricula, degree programs, accreditation criteria, and a code of ethics—is really impressive.*

– Dr. Robert L. Cannon, Distinguished Professor Emeritus of Computer Science, University of South Carolina

*Early in my career, I became involved in the SEI's Working Group on Software Engineering Education and Training via a conference presentation. The working group supported the birth and initial maturation of the undergraduate software engineering programs offered in the U.S. by providing a venue for software engineering educators to collaborate to provide needed structure and organization to the BSSE degree. The working group has also raised the visibility of software engineering as a discipline and the importance of SE education via conference presentations and publications.*

– Dr. Heidi Ellis, Associate Professor, Western New England University

### 3.3.3 The Consequence: Undergraduate Software Engineering Programs Established

In the past decade, the work of the WGSEET, the ACM and IEEE-CS, ABET, and the SEI have had a significant influence on the quantity and quality of undergraduate software education. ABET established a Software Engineering Program Criteria and have since accredited 27 programs under these criteria. The ACM and IEEE-CS developed the *Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering* (SE 2004) [ACM 2004], which provides curriculum development and pedagogy guidelines, a curriculum body of knowledge, example course descriptions, and curriculum architectures. The work of the WGSEET was a major influence on SE 2004, and several members of the WGSEET were part of the ACM/IEEE-CS task force. Other influences were the *Guide to the Software Engineering Body of Knowledge* [Bourque 2004] and an earlier SEI technical report on the software engineering body of knowledge [Hilburn 1999]. The ACM and IEEE-CS developed a *Software Engineering Code of Ethics and Professional Practice* [ACM 1999], which has influenced both curriculum development and accreditation planning. The

SEI adapted the Personal Software Process (PSP) and the Team Software Process (TSP) for academic use, and supported delivery of a series of summer workshops for faculty that taught faculty PSP and TSP techniques and engaged them in software process curriculum design [Hilburn 2002].

Most recently, the SEI has managed a project for the Department of Homeland Security to improve the state of software assurance practices by providing curriculum guidance in software assurance principles, methods, and practices (http://www.cert.org/mswa/).

### 3.3.4 The SEI Contribution

The SEI was a key player in the advancement of undergraduate software engineering education. The SEI and the WGSEET acted as software engineering education catalysts, providing forums for communication about education issues and supporting and motivating development of course modules, curriculum guidance, position papers, and technical reports [Bagert 1999b, Ford 1994, 1996, Hilburn 1999, Mead 2010, 2011, Shaw 2005]. While much of the work was accomplished by the stakeholders, it was the SEI impetus and leadership that accelerated the activity.

### 3.3.5 References

[ACM 1999] ACM/IEEE-CS Joint Task Force on Software Engineering Ethics and Professional Practices. *Software Engineering Code of Ethics and Professional Practice, Version 5.2*. ACM, 1999. http://www.acm.org/serving/se/code.htm

[ACM 2001] ACM/IEEE-CS Joint Task Force on Computing Curricula. *Computing Curricula 2001: Computer Science*, Final Report, December 15, 2001. ACM, 2001. http://www.acm.org/education/education/education/curric_vols/cc2001.pdf

[ACM 2004] ACM/IEEE-CS Joint Task Force on Computing Curricula, Software Engineering. *Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*. ACM, 2004. http://www.acm.org/education/curricula.html

[Bagert 1999a] Bagert, D.; Hilburn, B.; Hislop, G.; Lutz, M; & McCracken, M. "Guidance for the Development of Software Engineering Education Programs." *The Journal of Systems and Software 49*, (1999): 163-169.

[Bagert 1999b] Bagert, Donald; Hilburn, Thomas; Hislop, Gregory; Lutz, Michael; McCracken, Michael; & Mengel, Susan. *Guidelines for Software Engineering Education Version 1.0* (CMU/SEI-99-TR-032). Software Engineering Institute, Carnegie Mellon University, 1999. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=13565

[Bagert 2008] Bagert, D. J.; Port, D. N.; & Saideian, H. "Software Engineering Education, Training, and Research: The Legacy of Nancy Mead," 238-243. *Proceedings of the 21st IEEE Conference on Software Engineering Education and Training, 2008 (CSEET '08). April 14-17, 2008.* 238-243（doi: 10.1109/CSEET.2008.34).

[Bourque 2004] Bourque P. & Dupuis R., eds. *Guide to the Software Engineering Body of Knowledge*. IEEE CS Press, 2004. http://www.swebok.org/

[Ford 1994] Ford, G. *A Progress Report on Undergraduate Software Engineering Education* (CMU/SEI-94-TR-11). Software Engineering Institute, Carnegie Mellon University, 1999. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=12183

[Ford 1996] Ford, G. & Gibbs, N. E. *A Mature Profession of Software Engineering* (CMU/SEI-96-TR-004). Software Engineering Institute, Carnegie Mellon University, 1996. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=12515

[Gibbs 1989] Gibbs, N. E. "The SEI Education Program: The Challenge Of Teaching Future Software Engineers." *Communications of the ACM 32*, 5 (May 1989): 594-605.

[Hilburn 1999] Hilburn, Thomas; Hirmanpour, Iraj; Khajenoori, Soheil; Turner, Richard; & Qasem, Abir. *A Software Engineering Body of Knowledge Version 1.0* (CMU/SEI-99-TR-004). Software Engineering Institute, Carnegie Mellon University, 1999. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=13359

[Hilburn 2002] Hilburn T. & Humphrey, W. "Teaching Teams." *IEEE Software 19,* 5 (September 2002): 72-77.

[Hilburn 2003] Hilburn, T.; Duley, D.; Hislop, G.; & Sobel, A. "Engineering an Introductory Software Engineering Curriculum," 99-106. *Proceedings of the 16th Conference on Software Engineering Education and Training (CSEET)*. Madrid, Spain, March 20-22, 2003. IEEE, 2003.

[Mead 2010] Mead, Nancy; Hilburn, Thomas; & Linger, Richard. *Software Assurance Curriculum Project Volume II: Undergraduate Course Outlines* (CMU/SEI-2010-TR-019). Software Engineering Institute, Carnegie Mellon University, 2010. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9543

[Mead 2011] Mead, Nancy; Hawthorne, Elizabeth; & Ardis, Mark. *Software Assurance Curriculum Project Volume IV: Community College Education* (CMU/SEI-2011-TR-017). Software Engineering Institute, Carnegie Mellon University, 2011. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=10009

[Shaw 2005] Shaw, M., ed. *Software Engineering for the 21st Century: A Basis for Rethinking the Curriculum* (CMU-ISRI-05-108). Carnegie Mellon University, 2005. http://www.cs.cmu.edu/~Compose/SEprinciples-pub-rev2.pdf

## 3.4 Software Assurance Curriculum for Colleges and Universities

### 3.4.1 The Challenge: Demand for Software Assurance Expertise

Software plays a critical and central role in our personal lives and in the workplace, but it often has availability, reliability, safety, and security problems. The Bureau of Labor Statistics' *Occupational Outlook Handbook 2010-2011 Edition* [BLS 2011] highlights the need for software assurance expertise. It states,

> *Concerns over "cybersecurity" should result in the continued investment in software that protects computer networks and electronic infrastructure. The expansion of this technology over the next 10 years will lead to an increased need for software engineers to design and develop secure applications and systems, and to integrate them into older systems.*

Unfortunately, there are too few experienced professionals with the breadth and depth of software assurance knowledge that is essential to meet today's needs. In the future, the demand will greatly exceed the supply of skilled engineers—unless action is taken now.

### 3.4.2 A Solution: Educate Future Practitioners

DHS sponsored the SEI to build a model curriculum for software assurance education and define strategies to implement it. DHS had already worked with the SEI on the Build Security In website (https://buildsecurityin.us-cert.gov/) and recognized that the principle of building security in at the start had to extend to the workforce. In response to the DHS request, the SEI developed a curriculum model for a master of software assurance (MSwA) degree [Mead 2010a]. The SEI curriculum developers surveyed industry executives to understand their needs and worked with the academic community for contributions, refinement, and feedback on the curriculum. In particular, the SEI collaborated with faculty from Embry-Riddle Aeronautical University, Monmouth University, and Stevens Institute of Technology. Stevens was the first to implement a track corresponding to the curriculum.

The curriculum presents a body of knowledge that faculty can use to create a master of software assurance degree program, either as a stand-alone program or as a track in existing software engineering and computer science programs. The SEI and its collaborators developed syllabi for all nine courses [Mead 2011a], a bibliography of resources, and an overview seminar.[10] The seminar was first presented as a conference workshop [Mead 2010c]. The SEI also produced a podcast describing the curriculum [Mead 2010d]. In 2011, the curriculum was recognized by the IEEE and the ACM professional societies as a model curriculum for a master's degree program in software assurance, a significant achievement.

10 All materials are available on the CERT website: http://www.cert.org/curricula/software-assurance-curriculum.cfm.

The curriculum work was extended to address the educational needs at the bachelor and associate degree levels, drawing on the body of knowledge defined in the master's curriculum. The bachelor's level program [Mead 2010b] defines seven courses that could fit into another degree program, such as software engineering or computer science. These courses provide students with fundamental skills for either entering the field directly or continuing with graduate-level studies. The associate's degree program [Mead 2011b] also provides fundamental skills students need for further undergraduate-level work; those with prior undergraduate technical degrees can become more specialized in software assurance.

In 2013, the SEI developed on-demand software assurance training for executives and a software assurance competency model [Hilburn 2013]. The executive training includes slide sets and videos featuring experts from government and academia. The competency model helps organizations and individuals determine software assurance competency across a range of knowledge areas and includes a framework for adapting the model to an organization's particular domain, culture, or structure.

### 3.4.3 The Consequence: More Well-Qualified Software Assurance Professionals

The SEI software assurance curriculum helps meet the demand for professionals with a breadth and depth of software assurance knowledge. SEI materials ease the way for universities and colleges to implement software assurance programs and tracks. As of fall 2012, software assurance courses are in place at Carnegie Mellon University, Rochester Institute of Technology, Stevens Institute of Technology, University of Detroit Mercy, University of Houston, and the U.S. Air Force Academy; others are in progress. In addition, (ISC),[2] a training organization, has mapped its courses to the SEI curriculum.

Graduates of programs based on the SEI curriculum bring to a hiring organization the skills to deal with the security and quality of software systems in a comprehensive way. They can take responsibility for software assurance at an organizational level, in both its business and technical aspects. The graduates can apply assurance concepts to management, assurance assessment, people, and processes; understand how system requirements and specifications meet business needs; and assess software quality and security at a technical-design level and determine

**The View from Others**

*In our recent comprehensive curriculum review, we knew that we needed to enhance the focus on software quality assurance across our computer science courses. Most fortunately, we turned to the experts. The leadership, expertise, and resources provided by SEI with their software assurance curriculum hit right on target, immensely improving our ability to produce future Air Force officers prepared to defend and securely operate in cyberspace. We owe a great thanks to SEI, DHS, and the software assurance community that they have formed and fostered.*

– Steven Hadfield, Associate Professor and Curriculum Chair, Department of Computer Science, U.S. Air Force Academy

*These are terrific. They are a great contribution to curricula guidance generally and I am sure will be widely welcomed. Moreover they are particularly topical and relevant and I hope computing educators will take note.*

– Andrew McGettrick, Chair of the ACM Education Board and Council, and Associate Editor of the *Computer Journal* [SEI 2013]

how they relate to business needs. Most importantly, they can serve as a focal point for integrating assurance activities across the organization. They can perform almost any role in the organizations' software development lifecycle, including requirements engineer, software architect, test engineer, and project manager, or roles in software maintenance, operation, and acquisition.

The long-term goal of the SEI software assurance curriculum work is an increase in the reliability and security of software. Acquirers in government and industry, as well as consumers, have greater assurance that the software they buy will behave as expected.

### 3.4.4 The SEI Contribution

Many colleges and universities had degree programs in areas such as software engineering, computer science, and information security; but they lacked programs in software assurance. The SEI MSwA curriculum is the first created specifically for software assurance. The unique contributions of the SEI software assurance curricula are that they include both services and software and both acquisition and development. They also include business aspects, ensuring that graduates understand the relation of technology and business needs. The quality of the SEI work was recognized by the IEEE Computer Society and the ACM. A press release by the organizations states that this formal recognition signifies to the educational community that the MSwA reference curriculum is suitable for creating graduate programs or tracks in software assurance [IEEE 2010].

### 3.4.5 References

[BLS 2011] Bureau of Labor Statistics. *Occupational Outlook Handbook, 2010-11 Edition.* Bureau of Labor Statistics, 2011. http://www.bls.gov/ooh

[Hilburn 2013] Hilburn, Thomas; Ardis, Mark; Johnson, Glenn; Kornecki, Andrew; & Mead, Nancy. *Software Assurance Competency Model* (CMU/SEI-2013-TN-004). Software Engineering Institute, Carnegie Mellon University, 2013. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=47953

[IEEE 2010] Institute of Electrical and Electronics Engineers. "Computer Society Recognizes Master of Software Assurance Curriculum" (press release). December 8, 2010. http://www.computer.org/portal/web/pressroom/20101213MSWA

[Mead 2010a] Mead, Nancy; Allen, Julia; Ardis, Mark; Hilburn, Thomas; Kornecki, Andrew; Linger, Richard; & McDonald, James. *Software Assurance Curriculum Project Volume I: Master of Software Assurance Reference Curriculum* (CMU/SEI-2010-TR-005). Software Engineering Institute, Carnegie Mellon University, 2010. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9415

[Mead 2010b] Mead, Nancy; Hilburn, Thomas; & Linger, Richard. *Software Assurance Curriculum Project Volume II: Undergraduate Course Outlines* (CMU/SEI-2010-TR-019). Software Engineering Institute, Carnegie Mellon University, 2010. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9543

[Mead 2010c] Mead, Nancy & Ingalsbe, Jeff. "How to Get Started in Software Assurance Education." Conference on Software Engineering Education and Training (CSEET), Pittsburgh, PA, March 8-12, 2010.

[Mead 2010d] Mead, Nancy; Hilburn, Thomas; & Linger Richard. “Software Assurance: A Master’s Level Curriculum” (podcast). Software Engineering Institute, Carnegie Mellon University, 2010. http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=34734

[Mead 2011a] Mead, Nancy; Allen, Julia; Ardis, Mark; Hilburn, Thomas; Kornecki, Andrew; & Linger, Richard. *Software Assurance Curriculum Project Volume III: Master of Software Assurance Course Syllabi* (CMU/SEI-2011-TR-013). Software Engineering Institute, Carnegie Mellon University, 2011. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9981

[Mead 2011b] Mead, Nancy; Hawthorne, Elizabeth; & Ardis, Mark. *Software Assurance Curriculum Project Volume IV: Community College Education* (CMU/SEI-2011-TR-017). Software Engineering Institute, Carnegie Mellon University, 2011. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=10009

[SEI 2013] Software Engineering Institute, 2012 Year in Review, Carnegie Mellon University, 2013. http://resources.sei.cmu.edu/asset_files/AnnualReport/2013_001_001_46024.pdf

## 3.5 Survivability and Information Assurance Education for System Administrators

### 3.5.1 The Challenge: Adapting System Administration to the Unexpected and to Business

The Department of Defense, federal agencies, and industry organizations rely on networked systems that use fast-changing technology. Their reliance makes them more vulnerable to attacks, prompting system administrators to seek new approaches to computer and network security. Not only does technology change frequently, but it does not always work correctly. The network and system administrators who are most effective are the ones who can react to changes and other unusual situations—and can actively support the business mission. However, system administrators are typically trained to operate the technology following standard procedures; they aren't necessarily prepared to deal with the unexpected and rarely connect the technology to the fact that the business has a mission and that it has risk as part of its operations, along with policies and procedures and governance. They lack the perspective that an equipment failure or compromise could cause the business to fail, suffer financially, and lose credibility with the public. In addition, most system administrators inherit an existing network, yet few are taught how to analyze, maintain, and grow that type of network. Thus, they are not prepared to handle existing computer systems and network infrastructure components that, for example, are already (mis-) configured or have already been attacked, and for which documentation is misleading, incorrect, or nonexistent. It is important to educate system and network administrators about how to understand that network and positively participate in its management, all while keeping the mission and constraints of the business in focus.

### 3.5.2 A Solution: Survivability and Information Assurance Curriculum

In response to the U.S. National Guard's need for community college-level training for its system administrators, the SEI developed a four-semester course curriculum, with a capstone project, in survivability and information assurance (SIA). This curriculum offers a problem-solving methodology built on 10 key SIA principles that are independent of specific technologies. For example, one principle is survivability—ensure the business mission can survive in the face of attacks and breakdowns. Two others concern data: (1) Data is not just the information created by an application; the application and the operating system that runs the application are also data, and (2) it is essential to identify all data and then protect it appropriately, putting the most and best safeguards on the most critical data. Another principle stresses communication skills as critical.

The SIA curriculum was designed for experienced system and network administrators, ideally those who are receptive to the idea of technology supporting the business mission or already have a business sense that can be honed. The curriculum includes teaching materials, labs, instructor and student workbooks, and exams for all four courses, as well as material and instructions for the capstone exercise, in which students must make technological changes within the constraints of business considerations and articulate their decisions to a "manager" [Rogers 2006a, Rogers 2006b]. In 2006, the National Guard gave the SEI permission to distribute the courses through the

CERT website.[11] Free downloadable materials enabled faculty to start teaching SIA quickly. The materials are so extensive that faculty could use them as "turn-key" courses, or educational institutions could adapt them to suit their needs. Also, the materials are not only appropriate for two-year colleges but also for four-year institutions; portions are applicable at the graduate level. A read-only ("general") version of course materials is available to students and other system administrators.

### 3.5.3 The Consequence: System Administrators Who Support the Business Mission

System administrators make the connection between technology and business mission, using technology to help the business operate effectively and efficiently. They understand how a specific piece of technology fits into the overall business, its contribution to that business, and its importance. Their understanding of business and risk issues enables them to sustain and improve the enterprise's functionality and to add new functionality without a negative impact on the business mission. They are able to communicate with business decision-makers to explain, "This is what we're doing; this is why we're doing it; and this is why it makes business sense." They are able to answer questions such as "What's the impact on business? What are the metrics of performance? What does this cost? What's the benefit? Are we avoiding costs; are we reducing costs?" They help the enterprise networks to be better able to survive in an increasingly internet-oriented world.

### 3.5.4 The SEI Contribution

The SIA curriculum has filled a gap in system administrator education. It is a practical and realistic curriculum that layers skills training on a firm educational foundation and presents new ideas and new approaches to many of the traditional tasks of system administrators. The quality of the material and credibility of the SEI, its well-known CERT program, and Carnegie Mellon enable faculty to use the course material with confidence. The SEI influence is widespread. As of 2012, access to the faculty version of the curriculum was granted to 382 qualified faculty members representing 235 colleges and universities located in 43 U.S. states, the District of Columbia, and one Canadian province. The general, read-only version was downloaded 3,287 times by groups in 130 countries: 862 by organizations, 744 by educational institutions, 674 by government agencies, and 151 by others. The general version was also downloaded 856 times for personal use.

Others contributed to the influence of the SIA curriculum. The Regional Center for Systems Security and Information Assurance (CSSIA)[12] has mapped the SIA courses to two of the national standards for security-related training[13] and found that the courses meet approximately 95 percent of the standards' objectives.

---

11 Curriculum and lab overviews, and downloadable materials can be found at http://www.cert.org/curricula/sia-curriculum.cfm.

12 CSSIA is funded by the National Science Foundation.

13 *NSTISSI 4011 National Training Standard for Information Systems Security (INFOSEC) Professionals* , Committee on National Security Systems, (https://www.cnss.gov/CNSS/openDoc.cfm?U5Wzl9pAb0QOxSawdnQmTA==)1994 and *CNSS 4013 National Information Assurance Training Standard for System Administrators,* Committee on National Security Systems (http://www.scis.nova.edu/documents/cnssi_4013.pdf)

The SEI curriculum and others' contributions to it help increase the number of system administrators prepared to operate systems and networks that support business and government missions effectively and efficiently.

### 3.5.5 References

[Rogers 2006a] Rogers, Lawrence. "The CERT Survivability and Information Assurance Curriculum: Education for First Defenders." *Proceedings of the 10th Colloquium for Information Systems Security Education*, University of Maryland, Adelphi, MD, June 2006.

[Rogers 2006b] Rogers, Lawrence. "The CERT Survivability and Information Assurance Curriculum." 18th Annual FIRST Conference. Baltimore, Maryland, June 25-30, 2006. http://www.first.org/conference/2006/papers/rogers-lawrence-slides.pdf

2004.*CNSS 4013 National Information Assurance Training Standard for System Administrators*, entry level (https://www.ecs.csus.edu/csc/iac/cnssi_4013.pdf).

## 3.6 Conference on Software Engineering Education and Training

### 3.6.1 The Challenge: A Forum for Software Engineering Education Advances and Collaboration

Although software engineering education had emerged in a variety of programs, including the Wang Institute, IBM Software Engineering Institute, Seattle University, and Texas Christian University, there was little effort towards establishing a consistent curriculum. Not only was there no curriculum, there was no established mechanism for exchanging information about advances related to the course content, the courses themselves, or the teaching experiences, and no standard way to encourage professional collaboration. If the SEI was to successfully "influence software engineering curricula development throughout the education community" (SEI Charter 1984), it needed to lead the establishment of such a mechanism.

### 3.6.2 A Solution: The Premier Conference on Software Engineering Education

In 1986, the SEI conducted an invitation-only workshop on software engineering education [Gibbs 1987]. One of the outgrowths of this workshop was the first Conference on Software Engineering Education (CSEE), which took place in 1987 [Fairley 1989]. The conference was sponsored by the SEI with refereed papers and published proceedings by Springer Verlag. The major conference roles, including the chair and many of the program committee members, were SEI staff members. Along with the conference, a faculty development workshop was conducted to introduce new SEI curriculum modules and educational materials. In effect, the conference was established by the SEI to answer a need in the community, but it was largely run by SEI staff [Mead 2009]. One bold move in the early days of the conference series was to move the conference out of Pittsburgh and away from the SEI. After many years in Pittsburgh, the first venture to another location was the 1992 conference in San Diego. This particular conference, the 6th CSEE, was chaired by an SEI staff member and boasted an attendance of more than 200.

In the early 1990s, the CSEE also became the catalyst for an e-newsletter, the *Forum for the Advancement of Software Engineering Education* (FASE). Over the years, it provided informal information to software engineering educators on a periodic basis. It continues to provide announcements and articles to its audience, which is worldwide. The WGSEET was also an outgrowth of the conference. The working group met twice yearly between 1995 and 2000, producing a number of education reports and other artifacts [Mead 2009].

The conference transitioned from an SEI conference to an IEEE conference in 1996. This was a substantial step, as it involved replacing the events planning staff at the SEI with volunteers and securing IEEE sponsorship. It also meant that the conference would ultimately have to function as an independent financial entity, without support from the SEI. The SEI Education Program director was the first steering committee chair. She recruited steering committee members and developed the first formal charter for the conference. Several steering committee chairs have subsequently been appointed. Since this transition, the proceedings have been published by IEEE, and the conference chairs have been volunteers outside the SEI. With more community involvement, the program committee became much more diverse.

The CSEE conference series also broadened its focus to include a visible component on training; as of 1997, it became known as the Conference on Software Engineering Education and Training (CSEET). This addition reflected not so much a change of emphasis as recognition of the fact that education and training have always gone hand-in-hand. Starting in 2003, the conference has been periodically held in international locations, including Spain, Canada, Ireland, India, and China. This reflects the greater penetration of software engineering education in the international community. On occasion, CSEET has been co-located with other software engineering and education conferences, including the International Conference on Software Engineering (ICSE), the Innovation and Technology in Computer Science Education Conference (ItiCSE), and the ACM Special Interest Group on Computer Science Education (SIGCSE).

The CSEET has since evolved to include the ASEET, the Academy for Software Engineering Educators and Trainers. This addition to the conference supports the idea of mentoring as a way of growing the skills that are needed for professional software engineering education. In 2008, there were around 30 participants, and the speakers included leading software engineering educators as instructors. ASEET has continued to be part of the CSEET conference since then.

**The View from Others**

*CSEET has had a significant impact on the advancement of software engineering education and training. It has provided a forum for educators and software engineering professionals to interact about the preparation for software engineering practice: reporting on the state of the field, presenting new and innovative approaches, sharing experiences, and debating the best way forward.*

– Tom Hilburn, Embry-Riddle Aeronautical University

*CSEET has served as a focus for discussions that have led to such important milestones as the IEEE/ACM SE2004 undergraduate curriculum, and the GSwERC Graduate Software Engineering Reference Curriculum.*

– Tim Lethbridge, University of Ottawa

### 3.6.3 The Consequence: Growth of Conferences and Tracks on Software Engineering Education

The number of conferences and conference tracks related to software engineering education has grown. The ACM Special Interest Group on Computer Science Education has long sponsored a conference series [SIGCSE 2008] that has contained software engineering content from time to time. The ITiCSE conference [ITiCSE 2008] has also contained software engineering content. The series of Conferences on the Teaching of Computing (CTC) in Ireland had a considerable amount of software engineering content. More recently, ICSE has included a software engineering education track [ICSE 2008]. In 2000, a Carnegie Mellon professor presented a discussion of the future of software engineering education as part of a special track at ICSE [Shaw 2000]. Frontiers in Education (FIE) has always included software engineering content.

### 3.6.4 The SEI Contribution

Based on the results of the initial SEI workshop on software engineering education, the CSEE conference series was founded under SEI leadership. At the time, CSEE was the only conference focused on software engineering education, and it helped the software engineering education community to communicate, improve, and evolve. The conference later transitioned to a community-

based conference sponsored by IEEE and was broadened to specifically include training, thus being renamed CSEET. The conference has subsequently been held in a mix of domestic U.S. and international locations, and has been occasionally co-located with other major conferences. There were many additions and outgrowths from the conference, including faculty development workshops, FASE, WGSEET, and ASEET.

This is a typical example of how the SEI initially addresses a need with its own resources and with heavy participation of experts in the field. After the initial need has been met, the SEI finds an established mechanism to carry on the work and, while continuing its participation in the activity, withdraws to work on other important areas. Indeed, at this point, it is likely that most participants are unaware of the SEI's early contribution at the critical point.

### 3.6.5 References


[Fairley 1989] Fairley, R. E. & Freeman, P., eds. *Issues in Software Engineering Education-Proceedings of the 1987 SEI Conference on Software Engineering Education*. Monroeville, PA, May 30-June 1, 1987. Springer Verlag, 1989 (ISBN 0-387-96840-7).

[Gibbs 1987] Gibbs, N. E. & Fairley, R. E. *Software Engineering Education: The Educational Needs of the Software Community*. Springer Verlag, 1987 (ISBN 0-387-96469-X).

[ICSE 2008] *Proceedings of the 30th International Conference on Software Engineering*. Leipzig, Germany, May 10-18, 2008. ACM Publications, 2008.

[ITiCSE 2008] Amillo, J; Laxer, C.; Menasalvas Ruiz, E.; & Young, A., eds. *Proceedings of the 13th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE 2008)*. Madrid, Spain, June 30-July 2, 2008. ACM, 2008 (ISBN 978-1-60558-078-4).

[Mead 2009] Mead, N. R. "Software Engineering Education: How Far We've Come and How Far We Have to Go." *Journal of Systems and Software* (2009) (doi:10.1016/j.jss2008.12.038).

[Shaw 2000] Shaw, M. "Software Engineering Education: A Roadmap," 371-380. *Proceedings of the International Conference on Software Engineering (ICSE 2000)*, Future of Software Engineering Track, Limerick, Ireland, June 4-11, 2000. http://www.sigmod.org/dblp/db/conf/icse/future2000.html#Shaw00

[SIGCSE 2008] *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*. Portland, OR, March 12-15, 2008. ACM, 2008.

## 3.7 CMU Master of Software Engineering Program

### 3.7.1 The Challenge: The Need for a Strong Academic Software Engineering Program

As interest in developing new educational programs for software engineering grew in the 1980s, one problem was the absence of software engineering programs at major universities. Even with the SEI-led design of a recommended curriculum, some schools were reluctant to establish a new graduate degree until they had seen leadership from the top programs in computer science.

### 3.7.2 A Solution: Creation of the CMU Master of Software Engineering Program

In 1988, the dean of the CMU School of Computer Science (SCS) asked a group of faculty from the SCS and senior staff from the SEI to create a graduate program in software engineering. The SEI had just published a recommended curriculum for master's degrees in software engineering [Ardis 1988]. Carnegie Mellon would implement a version of that program, but with a unique component: a design studio modeled after those used in schools of architecture and design [Tomayko 1991]. The dean liked the way students worked informally with their peers and faculty advisors in the CMU School of Architecture; the master-apprentice model seemed as appropriate for learning software design as it did for other types of design.

The CMU Master of Software Engineering degree was a 16-month full-time program consisting of six required core courses, seven electives, and a four-term design studio [Gibbs 1990]. The program also included a two-semester weekly seminar that enabled students to improve their writing and presentation skills. Students from local industry could also attend part time, and all students were expected to have software development experience before entering the program. Many of the early students worked at the SEI or in local industry.

Initially almost all the core courses were taught by SEI staff in the video studio classroom at the SEI. This allowed the SEI to distribute videotaped lectures to educational partners interested in adopting those courses at their campuses.[14] The idea was to reduce the high startup cost of teaching new courses, but with the goal of training the faculty to eventually teach those courses on their own.

**The View from Others**

*My career now has a potential that I never before dreamed of. When I speak in meetings or conferences about software, people listen.*

– Anthony J. Lattanze, USAF. Andrews AFB, CA, MSE 1996 [CMU 1996a]

*The MSE was a great experience! The top of the line courses and the access to the SEI resources and documentation, combined with the real-world hands-on experience of the studio was the perfect preparation for my current job.*

– Alejandro Danylyzyn, Senior Consultant, Management Consulting/IT Practice, Deloitte & Touche Consulting Group, MSE 1996 [CMU 1996b]

14 Yodannis, J.; Hallman, H.; & Ardis, M. *An Assessment of the Pilot-Course Offering for the Video Dissemination Project* (CMU-SEI-88-MR-8). Software Engineering Institute, Carnegie Mellon University, 1988.

The design studio was also implemented initially within the SEI facility. Students worked on year-long projects for local industry and government organizations. Mentors for the students consisted of SEI staff, faculty from CMU, and visiting faculty from other universities. Core courses made use of problems in the design studio projects for examples and exercises.

As the CMU MSE program matured, faculty from SCS began to play a more active role. The core courses were redesigned to take advantage of faculty expertise and to emphasize cross-cutting techniques that could be applied throughout the software lifecycle [Garlan 1995]. Eventually, the program was moved completely into SCS on campus, although some courses are still taught by SEI staff.

### 3.7.3 The Consequence: New Academic Programs Established

After the CMU MSE program started, several other schools created their own graduate software engineering programs. In fact, the number of programs nearly doubled in the first three years after the CMU program started. Some of these new programs benefitted from the videotaped courses provided by the SEI. Others may have been inspired by the leadership of CMU. There are now more than 60 graduate software engineering programs in the U.S. and more than 40 undergraduate programs.

### 3.7.4 The SEI Contribution

The CMU MSE program was created as a joint effort of the School of Computer Science and the SEI, with equal numbers of staff from each organization involved in the initial design. Once that design was adopted, almost all the core courses and the design studio activities were taught by SEI staff for about five years. Gradually, more and more SCS faculty became involved in the program, and it was moved entirely into SCS. SEI staff members have continued to teach some courses and to serve as studio mentors.

### 3.7.5 References

[Ardis 1988] Ardis, Mark. “The Design of an MSE Curriculum.” Proceedings of the SEI Conference on Software Engineering Education. CSEE, Fairfax, VA, April 28-29, 1988. Published as Lecture Notes in Computer Science 327, Springer, 1988.

[CMU 1996a] Carnegie Mellon University. “Masters of Software Engineering at Carnegie Mellon” (Brochure), 1996.

[CMU 1996b] Carnegie Mellon University. “Masters of Software Engineering at Carnegie Mellon” (Brochure), 1996.

[Garlan 1995] Garlan, David; Brown, Alan; Jackson, Daniel; Tomayko, Jim; & Wing, Jeannette. “The CMU Master of Software Engineering Core Curriculum,” 65-86. *Proceedings of the 8th SEI Conference on Software Engineering Education*, New Orleans, LA, March 29-April 1, 1995. Springer, 1995 (ISBN 3-540-58951-1).

[Gibbs 1990] Gibbs, Norman E.; Ardis, Mark A.; Habermann, A. Nico; & Tomayko, James E. “The Carnegie Mellon University Master of Software Engineering Degree Program.” 152-154. *Proceedings of the Software Engineering Education Conference*, Pittsburgh, PA, April 2-3, 1990. Published as Lecture Notes in Computer Science 423, Springer, 1990.

[Tomayko 1991] Tomayko, James E. “Teaching Software Development in a Studio Environment.” *SIGCSE Bulletin 23*, 1 (March 1991): 300-303.

## 3.8 Executive Education Program

### 3.8.1 The Challenge: Effectively Managing Software Systems

With the success of the SEI academic curricula, a demand for executive education began to surface. Defense industry and DoD executives whose expertise was in areas other than software began to request executive education that would enable them to understand the issues associated with managing software-intensive systems.

### 3.8.2 A Solution: Executive Education Program

The SEI recruited a retired industry vice president and a retired flag officer to develop an executive education program [Pietrasanta 1987]. The executive education offerings were extremely popular with senior executives, and the Air Force adapted the material for a program called Bold Stroke for Officers at the rank of 0-6 and above. Recognizing the role software played in Desert Storm, the Army asked the SEI to offer a version of its executive education program to Army flag officers and Senior Executive Service (SES) civilians.

In the following years, the SEI continued to provide executive education to the DoD. The United States Army Strategic Software Improvement Program (ASSIP) funded SEI activities with the goal of dramatically improving the acquisition of software-intensive systems. One of the ASSIP-funded activities was the Army Senior Leader Education Program (SLEP). These annual programs were developed by the SEI for the Army and consisted of tutorials by SEI researchers about solving the software challenges encountered by the Army senior leaders.

The SEI continues to offer executive education programs for commercial organizations. For example, the SEI developed and hosted "Technovation," a week-long executive training lab specifically designed to support GE's Experienced Information Management Program (EIMP). Technovation participants represented a range of GE divisions and information technology roles. The executive training was delivered by experts from the SEI and guest lecturers from Carnegie Mellon faculty in areas such as robotics, interdisciplinary collaboration, team development, software architecture and product line development, service-oriented architecture, data mining, and cybersecurity [SEI 2012].

**The View from Others**

*Given that the complexity of systems is increasing exponentially, Lt. Gen. Ross Thompson…decided to make sure we can maintain these systems. The chief software architect (CSWA) will manage the software architecture to ensure best practices are being followed. This gives the PEO a better chance of overcoming system risks.*

– Robert Teri, U.S. Army Senior Software Acquisition Manager [SEI 2009, p. 20]

*Some of the biggest takeaways for us were concepts around interdisciplinary teams, creativity, and innovation. At GE we're always looking for new ways to be even more innovative, so this content was especially interesting and valuable.*

– Jennifer Cherry, Leader – IT Talent Development for GE, commenting on Technovation. [SEI 2012]

### 3.8.3 The Consequence: Improved Management by Executives

The SEI continues to develop and deliver executive education programs that meet the strategic needs of course customers. These educational programs support the growth of professionals and contribute to the evolution of these organizations' leadership and strategy.

For example, during the April 2009 Army Senior Leader Education Program (SLEP), Lt. Gen. Ross Thompson heard the director for the SEI Research, Technology, and System Solutions Program speak on the importance of architecture-centric practices. In this executive education session, attendees consistently named integration or interoperability as their main challenge. Thompson responded by declaring that appointment of a chief software architect (CSWA) would be mandatory for every Program Executive Office (PEO). The policy further mandates that the CSWA ensure consistent implementation of appropriate standards and processes [SEI 2009].

### 3.8.4 The SEI Contribution

The SEI has drawn upon the unique resources within the institute and within Carnegie Mellon to offer executive education that looks forward to new solutions. The SEI has continued to be sought out as a source for executive guidance and quality education. These programs have bolstered the adoption of SEI technologies and the success of our customers.

### 3.8.5 References

[Pietrasanta 1987] Pietrasanta, A. "Software Engineering Education in IBM," 5-18. *Proceedings of the 1987 SEI Conference on Software Engineering Education*, Monroeville, PA, April 30-May 1, 1987. (Published as *Issues in Software Engineering Education.* Springer Verlag, 1989, ISBN 0-387-96840-7).

[SEI 2009] "Army Requires PEOs to Appoint Chief Architect." 20-21. *Year in Review 2009.* Software Engineering Institute, Carnegie Mellon University, 2009. http://www.sei.cmu.edu/library/assets/2009_YIR.pdf

[SEI 2012] "SEI Bolsters GE Professional Development Curriculum with 'Technovation' Executive Training Lab." (Press Release) Software Engineering Institute, Carnegie Mellon University, 2012. http://www.sei.cmu.edu/newsitems/technovation-2011.cfm

## 3.9 Professional Training

### 3.9.1 The Challenge: Providing High-Quality Training to Software Practitioners

SEI research projects produced results that required training project stakeholders to successfully transition the solutions. The training work products could, in turn, support the adoption of research by an audience beyond the SEI project stakeholders. Professional training would enable the SEI to disseminate knowledge on a large scale and build a solid base of competency among software engineering practitioners.

### 3.9.2 A Solution: Quality Assurance for SEI Training Products

To meet this dissemination need, the SEI established the means to provide high-quality professional education and training. The SEI produced a series of instructor-led training courses that were offered both in the SEI and elsewhere. SEI courses are developed and taught by technical staff members, and effectively disseminate mature SEI solutions to pervasive problems.

The Education and Training Review Board (ETRB) was established in 1990 to provide a mechanism for quality assurance for all SEI education and training materials produced in-house. The ETRB reviewed and approved each instructional product at three points during the development process: proposal, design, and product completion.

A supplemental review was required if the product underwent significant revisions after final approval and product release. The ETRB comprised six representatives (including the ETRB chair), one from each of the following SEI areas:

1. education
2. instructional design
3. software technology
4. product planning
5. program development
6. technical communications

Over time, the ETRB members realized that they were no longer effectively serving the changing needs of the SEI's organizational structure and culture, and the board was disbanded in 1997. The ETRB process guidelines were replaced by SEI Work Process 4.1, "Instructional Product Development," which was subsequently added to the list of SEI products covered by SP800-10, "Document Development." This standard practice has since evolved into a best practice for the SEI.

With these quality procedures in place, the catalog and popularity of SEI education and training continued to grow. In 1995, the SEI had 12 courses, and SEI instructors delivered 33 public course offerings and 31 customer on-site offerings. By 1997, the SEI had a catalog of 17 courses about process improvement, measurement, change management, and risk management. By 2001, the SEI had 34 courses, with most new courses addressing network and computer security.

In 2003, the SEI announced a new software architecture curriculum. Based on decades of experience in architecting software-intensive systems and supported by widely acclaimed practitioner

books in the SEI Addison-Wesley Series, this collection of six courses equips software professionals with state-of-the-art practices so they can efficiently design software-intensive systems that meet their intended business and quality goals.

The SEI established the Professional Development Center (PDC) in 2008. The PDC was staffed by education and administration professionals responsible for supporting and growing the SEI's flourishing professional education activity.

The SEI now offers courses for participants at many skill levels. Introductory courses are designed to provide a basic understanding of proven practices, while advanced courses train participants to measure the process capability of their own organizations or other organizations. Software managers and practitioners learn about proven techniques to increase profit and quality, adapt to change, build teams, mitigate risk, and improve process. The course offerings are held at SEI facilities in Pittsburgh, Pennsylvania, and Arlington, Virginia; online; and at other national and international venues.

### 3.9.3 The Consequence: SEI Results Move from Research into Practice

SEI professional courses help to move state-of-the-art technologies and practices from research and development into widespread use. The software engineering professionals from around the globe who attend SEI professional training courses broaden their skills and future career opportunities.

The content of SEI training continues to evolve based on new research. The SEI also continues to evolve the training delivery infrastructure to provide the best possible vehicles for the dissemination of SEI solutions.

### 3.9.4 The SEI Contribution

As the course catalogs grew, the SEI provided high-quality training that received consistently positive feedback from learners. SEI courses present the relevancy of SEI research to the individual and to his or her workplace. Professional training has proven to be an effective vehicle for meeting the SEI's mission of transition.

The SEI's professional education and training activity also contributed to the growth of a market for software engineering professional training. Many commercial professional training providers now offer training based on SEI methods and tools.

## 3.10 Education and Training Delivery Platforms

### 3.10.1 The Challenge: Deliver Education and Training to Large, Geographically Dispersed Audiences

In 1984, not only were there very few academic programs in software engineering, there were very few software engineering courses and few faculty prepared to teach the subject matter. Software engineering was evolving, and it included subject matter (such as software specification, verification and validation, and project management) not normally covered in computer science or other disciplines. Although the subject matter was taught in various industrial continuing education programs for practicing engineers, it was not part of an accepted academic discipline [Ardis 2005].

Faculty who participated with the SEI in development of the model curriculum made clear that, while they had an intellectual grasp of the material, they did not have the experience to translate that material into lectures and courses. Commercial companies were training their practicing engineers on evolving methodologies specific to their respective companies, but this material was generally not visible to the academic community.

### 3.10.2 A Solution: Take Advantage of Changing Technologies

The SEI recognized the opportunity to accelerate the learning curve by using technology. The SEI could increase its support to other universities by videotaping the Carnegie Mellon courses and offering them to other colleges and universities. Recognizing that it was neither practical for SEI people to travel to other universities nor for university faculty from other universities to spend significant time at the SEI, the SEI constructed a video studio. The class lectures of six courses from the model curriculum selected by the CMU MSE program [Gibbs 1990] were delivered to CMU students in the studio. The video recordings of these lectures were then provided to other universities. Some universities showed the videos as the course offering. At other universities, faculty reviewed the videos to further their own understanding and then offered similar, tailored lectures to their students. Florida Atlantic University (FAU) had a unique partnership with the SEI and graduated a number of master of software engineering students as well as certificate holders—as a result of the partnership and the creative way in which FAU used the MSE videos and collaborated with the MSE faculty.

In subsequent years, the SEI has continued to supplement traditional classroom training by taking advantage of new technology to capture and disseminate training. The video studio, created to support the academic education courses, was subsequently used to record lectures by leaders in the software engineering community to provide insight into important developments, resulting in a stand-alone public video series. The video studio was also used to videotape many of the SEI continuing education courses.

As media moved beyond videotape, the SEI developed a catalog of asynchronous web-based, self-paced e-learning and broadcast virtual “live” online training. The SEI maintains an eLearning Portal, which is the platform for the development and delivery of SEI eLearning courses. This portal supports a wide variety of training formats to best fit the SEI content and learners’ preferences. Blended learning courses, especially the Personal Software Process (PSP) curriculum, use both an online and a classroom instructor to provide instruction. Other asynchronous online

courses use video, animation, graphics, tests, and recorded lectures to bring SEI instructional content to learners across the globe.

More recently, the SEI has been producing podcasts and webinars. The SEI website provides free access to the SEI Podcast Series, a collection of recordings of SEI experts discussing their research and field experience. The SEI Webinar Series provides a public forum for the latest research, best practices, and cutting-edge solutions developed at the SEI. These webinars feature SEI researchers discussing their work in a flexible format. Customers can view a webinar in real time and then participate in the live question and answer session that follows each presentation. Customers can also view webinar recordings later as streaming video. The SEI periodically hosts virtual events that offer to the public a series of interactive, live lectures by SEI experts. Webinars and virtual events regularly draw an audience of hundreds of learners interested in the latest developments of SEI research.

The SEI is keeping pace with learning trends with m-learning—mobile learning on the go—ensuring that online training resources are available on the learner's workstation, laptops, and tablets.

### 3.10.3 The Consequence: Software Engineering Is Adopted as a Discipline

There can be little doubt that software engineering would have become an accepted academic discipline had the SEI not undertaken the video studio. However, it is equally clear that the SEI was in large part responsible for accelerating the process. That a university with Carnegie Mellon's stature in computing undertook to offer a master's degree in software engineering, based on the model curriculum, and made videotapes of those courses available contributed significantly to early adoption by other colleges and universities. Certainly there are university programs that evolved independent of SEI influence, but the fact that the SEI model curriculum was endorsed by the IEEE and ACM demonstrates that the SEI's influence was a significant factor.

To maintain this positive influence on the maturation of software engineering, the SEI continues to evolve the channels of disseminating educational material to an ever-broadening audience. From early videotapes to the current eLearning portal, the SEI reaches a global audience.

### 3.10.4 The SEI Contribution

The SEI role was to accelerate the transition of material from industrial experience to the academic community. The idea of a video studio had already been demonstrated at other universities and in industry for other types of courses. The subject matter had already been tested in the academic community at the Wang Institute [Ardis 1987]. However, because of the SEI's position in a respected university, incorporating those ideas into a framework that made it accessible to other colleges and universities convinced faculty elsewhere that material could be taught effectively and had a sufficient academic foundation. This success led to a successful adoption of internet-enabled learning that continues to benefit a broad audience.

### 3.10.5 References

[Ardis 1987] Ardis, Mark. "The Evolution of Wang Institute's Master of Software Engineering Program." *IEEE Transactions on Software Engineering 13,* 11 (November 1987): 1149-1155.

[Ardis 2005] Ardis, Mark. “An Incomplete History of Master of Software Engineering Programs in the United States.” Presentation at the 15th Reunion of CMU MSE Program, August 6, 2005. http://personal.stevens.edu/~mardis/papers/MSEHistory.pdf

[Gibbs 1990] Gibbs, Norman E.; Ardis, Mark A.; Habermann, A. Nico; & Tomayko, James E. “The Carnegie Mellon University Master of Software Engineering Degree Program,” 152-154. *Software Engineering Education Conference*. Pittsburgh, PA, April 2-3, 1990. Published as Lecture Notes in Computer Science #423. Springer 1990.

## 3.11 Technology for Cyber Workforce Development

### 3.11.1 The Challenge: Training the Cyber Workforce in a Rapidly Changing World

Organizations are faced with the ongoing challenge of ensuring that their workforces have the most current knowledge, skills, and experiences to gain proficiency in areas relevant to their jobs. This challenge is particularly great for a cybersecurity workforce because industry trends, practices, and technologies are constantly changing, and attackers constantly seek new ways to circumvent security controls and infiltrate systems. Moreover, it is expensive for organizations to develop and execute scalable and realistic training events and to reach the large number of people who need the training. Traditional training models still use brick-and-mortar classrooms to provide infrequent instruction directed at individual students. This approach cannot keep up with the pace of change or provide effective mechanisms for the workforce to get the real-world experience it needs to operate in cyberspace. It also takes students away from their job duties and leads to lost productivity.

The Department of Defense has some additional, unique challenges. The dynamic nature of the internet threats, combined with the anonymity it allows, the absence of borders, and undefined jurisdictions and laws, make it challenging to define and achieve successful offensive and defensive operations. Although the DoD has made some gains in cyber defense technologies, less progress has been made in defining cyber operations; traditional training approaches are leaving U.S. cyber warriors under-trained, reactive, and at a tactical disadvantage. The DoD workforce cannot effectively train as it fights as part of normal operations, and service members and DoD civilians cannot use production networks for operational training. As a result of these limitations, U.S. national security is at risk [Gjelten 2010].

### 3.11.2 A Solution: Virtual Training Technology for Individuals and Teams

The SEI Virtual Training Environment (VTE) and Exercise Network (XNET) have provided cost-effective, up-to-date, and relevant training to more than 100,000 active, globally distributed users, meeting the unique training requirements of cybersecurity [Hammerstein 2010, USSS 2005]. In 2013 they evolved into the Simulation, Training, and Exercise Platform (STEPfwd) [CERT 2013].

All three technologies have elements in common: platform-independent, web-based delivery of video-captured instruction that presents a “read it, hear it, see it, do it, master it” progression of course lectures, demonstrations, and hands-on labs. Individual workforce members can access the training anytime, anywhere. The material is divided into modules they can fit into their work schedule. The lecture portion includes the same elements found in an instructor-led setting, such as slides, visuals of the instructor and other students, and demonstrations. It includes a script as another mode for learning, and a topic index and progress indicator that aid self-pacing [Hammerstein 2010].

VTE was the culmination of early SEI work in using technology for distance education and training, which started at the SEI’s inception in 1984. Initially, the SEI recorded lectures onto videotapes and mailed them to remote students. As standards for video on computers emerged, the SEI shifted to distributing lectures on CD and DVD-ROM. The SEI also developed a multimedia course for DARPA in 1993, in response to a request for training that was scalable and more interactive and engaging than videotape. The course development was expensive and too lengthy to be

practical for keeping content current. The following year, the SEI offered courses through the National Technological University. Distant viewers could watch a live class and interact through telephone and a precursor of chat. However, they preferred videotapes of the class to watch at their convenience in their offices; they valued self-pacing. The next SEI advance was Just-in-Time Learning (JITL), which was modular, quick, low cost, and scalable. It allowed both self-pacing and keeping content current. JITL was the direct precursor of VTE and contained many of the same elements. In 2005, JITL was ported to the web to become VTE.

XNET extended the capabilities of VTE's individual training by providing a virtual, isolated, and dedicated exercise environment that enabled teams to gain realistic experience outside of their operational networks. Development started in 2008, in response to the cyber training and evaluation needs of the DoD, to provide practical experience and evaluate the readiness of DoD personnel [SEI 2011]. SEI staff observed the DoD's difficulty in setting up cyber exercises while giving direct support for exercises at military academies. In addition, the real-world experience of two SEI staff members who were in the military gave them first-hand knowledge of the difficulty of providing essential training to military units. In XNET, trainees saw a replication of the systems they were using in real life. Using a standard web browser, a team could assemble in a "virtual room," regardless of members' physical location. The training events could scale from small teams to hundreds of globally dispersed participants.

The next advance was STEPfwd, first available in 2013 (https://stepfwd.cert.org/vte.lms.web). STEPfwd combines elements of VTE and XNET, while taking advantage of advances in technology, to provide continuous professional development for cybersecurity professionals. It includes individual on-demand training similar to the virtual classroom of VTE and the network environments similar to XNET for team development. In addition, network simulations can be built in real time. Both individuals and organizations can track progress.

**The View from Others**

*Exercises like Cyber Flag are important because they provide an assessment and a validation of how well U.S. Cyber Command can perform its real-world mission to operate and defend the DoD networks across the full range of cyber operations…*

*The exercise was successful because it enabled the command to integrate and synchronize joint warfighting efforts with the Service Cyber Components in a realistic scenario.*

– Air Force Col. George Lamont, on Cyber Flag [Johnson 2011]

*A superb, world-class event. I saw a complete cadre of cyber warriors so energized about fighting an extremely complex, realistic cyber threat scenario.*

– Gen. Jon Davis, deputy commander for U.S. Cyber Command, on Cyber Guard [Johnson 2012]

### 3.11.3 The Consequence: Unified Platform for Cyber Workforce Development

The DoD and federal and state agencies have a highly skilled workforce that can handle the challenges of a fast-changing cyber world. The workforce maintains essential proficiency even if it is globally distributed, and those in charge of training can accurately evaluate their personnel's mission readiness. Most importantly, DoD cyber warriors have an advantage over their adversaries. Individuals using STEPfwd (and, previously, VTE and XNET) technologies shift between the

training and job duties without one interfering with the other. Travel to classroom training is eliminated, and productivity is maintained. They gain essential cyber training that the SEI keeps current with the rapidly changing state of cybersecurity. Distributed teams using STEPfwd gain hands-on experience with the latest security threats and technology, and practice their skills against real-world scenarios. They work in an isolated environment that mimics their operational network and eliminates the setup and configuration time associated with hosting an exercise. The experience they gain through routine practice is a decisive factor in how effectively they respond in emergency situations and, for the DoD, combat operations. Effective and up-to-date cyber training prepares the cyber workforce to safeguard U.S. national security.

### 3.11.4 The SEI Contribution

The SEI shifted the paradigm away from the traditional classroom-based training models. VTE, XNET, and STEPfwd have been next-generation content delivery platforms with no equivalent in the private sector. The technology's flexibility enables the SEI to cost-effectively keep the content current with constantly changing attack techniques, industry trends, practices, and technologies.

The SEI is filling technology and methodology gaps to empower the DoD and government agencies to train and maintain the most capable and mission-ready cyber workforce in the world. More than 80,000 service members and civilians used VTE to meet the compliance requirements defined in the Department's 8570.1 Information Assurance Workforce Development program [SEI 2008]. VTE is also the primary training and certification mechanism for the DoD's Host Based Security System (HBSS) initiative. In 2012, VTE was transitioned to, and is now operated by, the federal government as part of its FedVTE program (https://www.fedvte-fsi.gov/).

XNET was used by several DoD organizations. Notably, it was the primary training platform/game space for U.S. Cyber Command's (CYBERCOM) Cyber Flag joint military cyber exercise and the Office of the Secretary of Defense's (OSD) International Cyber Defense Workshop. In the fall of 2013, STEPfwd was used for CYBERCOM's joint military showcase event, Cyber Flag 14-1. The SEI will continue to investigate new approaches to continuous operational training and evaluation that make knowledge, skills, cyber experience-building, and measurement globally accessible in highly collaborative, web-centric learning environments.

### 3.11.5 References

[CERT 2013] *Cyber Workforce Development.* CERT Program, Software Engineering Institute, Carnegie Mellon University, 2013.

[Gjelten 2010] Gjelten, Tom. "Cyberwarrior Shortage Threatens U.S. Security." National Public Radio (NPR), July 19, 2010. http://www.npr.org/templates/story/story.php?storyId=128574055

[Hammerstein 2010] Hammerstein, Josh & May, Christopher. *The CERT Approach to Cybersecurity Workforce Development* (CMU/SEI-2010-TR-045). Software Engineering Institute, Carnegie Mellon University, 2010. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9697

[Johnson 2011] Johnson, Col. Rivers. "U.S. Cyber Command Conducts Tactical Cyber Exercise." *Soundoff!* December 10, 2011. http://www.ftmeadesoundoff.com/news/9475/us-cyber-command-conducts-tactical-cyber-exercise/

[Johnson 2012] Johnson, Col. Rivers. “Cyber Guard Exercise Focuses on Defensive Cyberspace Operations.” U.S. Army, August 16, 2012. http://www.army.mil/mobile/article/?p=85786

[SEI 2008] “VTE Helps DoD Meet Remote Training Requirements and Cut Costs.” *2008 Year in Review.* Software Engineering Institute, Carnegie Mellon University, 2008: 11-12. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=30140

[SEI 2011] “CERT Exercise Network (XNET) Instrumental in Successful Test of Cyber Attack Readiness,” 30-31. *2011 Year in Review.* Software Engineering Institute, Carnegie Mellon University, 2011. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=30121

[USSS 2005] United States Secret Service, Criminal Investigative Division, USSS-CERT Liaison. “Virtual Training Environment (VTE) Pilot Study.” Pittsburgh, PA, 2005.

# 4 Management

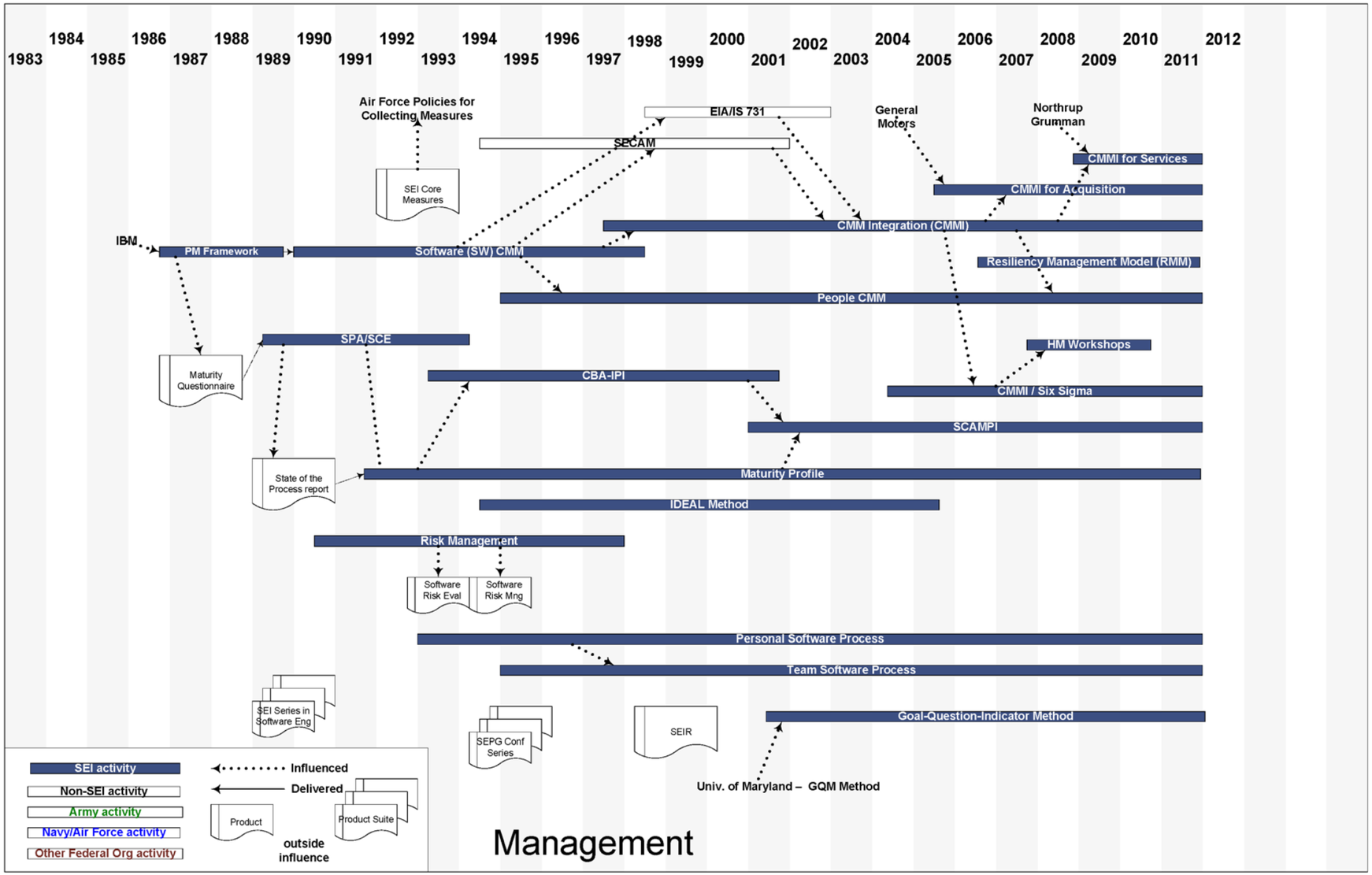


*Figure 5: Management Timeline*

## 4.0 Introduction to Management

When the SEI was established, the DoD and defense contractors generally understood that certain software engineering practices produced consistent results. Unfortunately, those practices were not documented or widely recognized. Software project planning and tracking, version control, configuration management, and quality management were not well understood in the context of the overall management of a software development project. To complicate matters, software tools supporting such activities were often home-grown, with idiosyncrasies peculiar to the organization. Often, the choice of tool would dictate how processes were defined and how they interacted. Nevertheless, there were examples of successful software development programs and a growing body of literature to support the concept that software could be managed by a process [Boehm 1981].

### 4.0.1 Management of the Software Process

The DoD identified the management of the software component of programs as a major problem area [DoD 1982], and the DoD STARS strategy envisioned a managed process well supported by automated tools [Druffel 1983]. Consequently, the SEI strategic plan identified the management of the software development process—by both the DoD program offices and the defense contractors—as a fundamental activity [Barbacci 1985]. Several companies, most notably IBM, had mature efforts aimed at management of the software development process that had proven effective [Humphrey 1985].

The SEI recruited a retiring executive from IBM who had been instrumental in creating and managing the IBM efforts and initiated work on the process management framework in 1986. Shortly thereafter, the Air Force program manager asked the SEI to conduct a study of "best practices." The SEI used this customer interest to drive the process management framework effort. Several workshops were conducted with leading software professionals in the DoD, defense industry, commercial industry, and academia to elicit a consensus on practices that consistently led to improved software development. Eighteen practices were identified. To help organizations determine how well their work stacked up against these practices, the SEI produced a Maturity Questionnaire [Zubrow 1994]. Response to this questionnaire was overwhelmingly positive, both from the DoD and the defense industry.

Initially, the questionnaire identified 18 key process areas and a five-level model of organizational maturity based on the implementation of the process areas. As the community began to adopt these ideas, they expressed the need for a more precise definition of the practices and the model. In response, the SEI developed the Capability Maturity Model for Software[15] [Paulk 1993], published a book, and created several reports on managing the software process to help community members evaluate their process management capability and proactively take steps to improve

15 Paulk, M. C.; Curtis, B.; Chrissis, M. B., et al. *Capability Maturity Model for Software* (CMU/SEI-91-TR-24). Software Engineering Institute, Carnegie Mellon University, 1991. No longer available.

[Humphrey 1989, Fowler 1990, Kitson 1993]. With the publication of an explicit model of software development practice, worldwide adoption grew each year for many years. The Software CMM thus launched the process improvement movement, and other CMMs eventually emerged from the SEI and others as communities recognized their value [Davenport 2005].

### 4.0.2 Support for Acquisition Offices

After the initial efforts to help contractors assess the maturity of their software processes and initiate process improvement activities, the Air Force program manager asked the SEI if there were a way that the acquisition offices could assess the maturity of their contractors. This request put the SEI in a serious dilemma. From the earliest days of the assessment work, the SEI made it clear that it must have the ability to protect the privileged information that it gathered in assisting defense contractors. Yet, on behalf of the DoD customer, the Air Force made the case that program offices needed some insight into their respective contractors. The SEI, therefore, began investigating the possibility of an assessment mechanism for use by government program offices. The Software Capability Evaluation (SCE) method was developed by the SEI, along with training and documentation to support its use by the DoD acquisition community [Averill 1993]. The SCE method was widely used in software-intensive systems acquisitions and provided an incentive for the use of the SEI's CMM to achieve improvements in both management and technical practices within the community that served the DoD. The current evolution of this work is considered a de facto standard for evaluating and improving process management in software and systems engineering.

### 4.0.3 Maturity Profile

While the models were useful for improvement and appraisal, initially there were no data available on the adoption of the model or the results of the appraisals. To address this need, the Community Maturity Profile was developed. It began with the publication of a report on assessments conducted through 1991. Then the SEI realized the community needed more information about the process maturity of the software engineering community. The SEI began the work of developing the processes, procedures, and infrastructure to allow the reporting of assessment results as a matter of routine by those performing them. In 1992, the SEI began publishing a quarterly report that included a summary and analysis of the accumulated assessment results. Providing this kind of updated information of value to the community motivated those conducting assessments to share their results with the SEI. Also, as more results were submitted, more analyses could be conducted; and the ideas for those analyses came from members of the community as well as from SEI researchers. Over time, the Community Maturity Profile became the principal source of information regarding the adoption of the CMM and CMMI and the state of the community in terms of process improvement [Zubrow 2003].

### 4.0.4 Expansion of Maturity Modeling

After the first Capability Maturity Model was created for use in software development, it became widely recognized as a powerful framework for understanding and improving processes in multiple disciplines. In response to strong community requests, the SEI produced maturity models for three other areas: the People CMM [Curtis 2003] for managing human assets, the Systems Engi-

neering CMM [SEI 1995] to describe the essential elements of an organization's systems engineering process, and the CERT Resilience Management Model for managing operational resilience [Caralli 2010].

### 4.0.5 The People CMM

The People CMM employs the same Process Maturity Framework of the SW-CMM to support the foundation of best practices for managing and developing an organization's workforce. Based on the best practices in fields such as human resources, knowledge management, and organizational development, the People CMM guides organizations in improving their processes for managing and developing their workforce. The People CMM helps organizations characterize the maturity of their workforce practices, establish a program of continuous workforce development, set priorities for improvement actions, integrate workforce development with process improvement, and establish a culture of excellence.

### 4.0.6 The Systems Engineering CMM (SE-CMM)

The SE-CMM describes the essential elements that must exist to ensure good systems engineering and provides a reference for comparing actual systems engineering practices against them. Good systems engineering is the key to success in market-driven and contractually negotiated market areas, and the SE-CMM provides a way to measure and enhance performance in that arena. It was designed to help organizations improve through self-assessment and guidance in the application of statistical process control principles. In conjunction with the model itself, a companion appraisal method exists, the SE-CMM Appraisal Method, for benchmarking the process capability of an organization's or enterprise's systems engineering function.

### 4.0.7 The CERT Resilience Management Model

In the late 1990s, the DoD faced a set of problems shared with organizations in every sector—U.S. federal government agencies, defense and commercial industry, and academia—arising from increasingly complex business and operational environments. These problems involved stress related to operational resilience, that is, the ability of an organization to achieve its mission even under degraded circumstances. The traditional disciplines of security, operational continuity, and information technology operations needed to be expanded to provide protection and continuity strategies for high-value services and supporting assets commensurate with these new operating complexities. The SEI recognized that the best practices of such organizational challenges could best be managed with a capability maturity model. Over the ensuing 10 years, the SEI engaged the relevant communities in evolving such a model, reflecting the best practices of such diverse organizations as the DoD, defense industry, commercial industries, and financial services, to evolve a maturity model that embodied best practices. This work culminated in the CERT Resilience Management Model that was released in 2010. It has been applied successfully to a wide range of problems, including assessing the capability of U.S. IT-based critical infrastructures to be resilient in the presence of an attack and building an incident management capability in developing nations.

### 4.0.8 Integration of Maturity Modeling

The Capability Maturity Model Integration (CMMI) effort was initiated to improve the usability of maturity models by integrating many different models into one framework. The best practices in CMMI more explicitly linked management and engineering activities to business objectives and incorporated lessons learned from several additional areas, including measurement, risk management, and supplier management. The software engineering, management, and measurement processes defined by CMMI have seen widespread adoption, with implementations in 74 countries on six continents.

CMMI models were developed in three specific areas: product and service development [CMMI 2010a], service establishment, management, and delivery [CMMI 2010b], and product and service acquisition [Bernard 2005]. Each CMMI model was designed to be used in concert with other CMMI models, making it easier for organizations to pursue enterprise-wide process improvement.

#### 4.0.8.1 CMMI for Development

The CMMI for Development (CMMI-DEV) covers activities for developing both products and services. Organizations from many industries, including aerospace, banking, computer hardware, software, defense, automobile manufacturing, and telecommunications, use CMMI-DEV. It includes practices that cover project management, process management, systems engineering, hardware engineering, software engineering, and other supporting processes used in development and maintenance.

#### 4.0.8.2 CMMI for Acquisition

The CMMI for Acquisition (CMMI-ACQ) provides guidance for applying CMMI best practices in an acquiring organization. Although suppliers can provide artifacts useful to the processes addressed in CMMI-ACQ, the focus of the model is on the processes of the acquirer. Best practices in the model focus on activities for initiating and managing the acquisition of products and services to meet the needs of customers and end users.

#### 4.0.8.3 CMMI for Services

The CMMI for Services (CMMI-SVC) model provides guidance for applying CMMI best practices in a service provider organization, with best practices covering the activities needed to provide quality services to customers and end users. CMMI-SVC integrates bodies of knowledge that are essential for a service provider and includes best practices from government and industry. It can be treated as a reference for the development of the service system that supports delivery of the service. In cases in which the service system is large and complex, the CMMI-DEV model can be effectively used to develop such a system.

### 4.0.9 Smart Grid Maturity Model: A New Approach for Utilities

The demand for electricity is projected to nearly double worldwide from 2009 to 2035, which could lead to significantly higher prices and blackouts. A smart grid could help to address these issues, but the transformation to smart grid is a major undertaking. The Smart Grid Maturity Model (SGMM) [SGMM 2011] supports utilities planning a move to smart grid use. It was placed

under SEI stewardship in 2009. Development of the SGMM began in 2007, when IBM formed a coalition of major utility companies to change the way power is generated, distributed, and used by adding digital intelligence to the system. The SGMM product suite now consists of the model itself, the Navigation Process of expert-led workshops and analysis, the Compass questionnaire-based assessment for determining maturity ratings and performance comparisons, Navigation Process training, and a program to license organizations and certify individuals to deliver the Navigation Process.

### 4.0.10 Characterizing Software Risks

As the SEI technical reputation grew, the DoD began requesting that the SEI conduct "red-teams" on programs that appeared to be in serious trouble. After one such exercise, the DoD program manager asked the SEI to provide a briefing of its findings. After the presentation, a senior DoD official in the audience issued a challenge: "It is clear that the SEI can convene a team of software experts and successfully assess the risks in a software system. Would it be possible for the SEI to capture that knowledge and present it in a way that knowledgeable people who are not experts can make a similar assessment without the SEI's help?"

The SEI launched a research effort to identify and offer mitigating strategies for software risks. The expectation was that by capturing and cataloging sources of software failures, DoD program offices and their contractors would have a basis for discussion about risk in their programs. The SEI produced several related risk management products that had the desired effect. The SEI risk approach became the de facto mechanism used by many program offices. One Navy program manager commented that the approach allowed him to identify a single, potentially catastrophic risk, which made the effort worthwhile.

As the SEI understanding of software-related risks matured, that understanding led to approaches that influenced several SEI products, including security, commercial off-the-shelf (COTS) software, architecture, and the Risk section of the CMMI.

### 4.0.11 Bringing Discipline to Software Development Activities

While the CMM had a substantial positive effect on the management system for software development [Herbsleb 1997], another significant step in quality was taken when the improvement process was extended to the people who actually do the work—the practicing engineers [Humphrey 1995]. The Personal Software Process (PSP) was built on the principle that every engineer who works on a software system must do quality work to produce a quality system. The development of the PSP began with the application of CMM principles to writing small software programs. Further refinement led to a disciplined personal framework that allows engineers to establish and commit to effective engineering and management practices for their software projects.

Skilled engineers and defined processes are essential for developing software-intensive products, but effective engineers must also be able to work on teams. The Team Software Process (TSP) was developed to help development teams establish a mature and disciplined engineering practice that produces secure, reliable software in less time and at lower costs [Humphrey 2000]. The TSP has been applied in small and large organizations in a variety of domains with documented re-

sults, including productivity improvements of 25 percent or more, testing costs and schedule reductions of up to 80 percent, and cost savings of 25-50 percent per software product release [Davis 2003].

### 4.0.12 Measurement and Analysis

While some early software projects used systems for predicting software costs, measurement was done in different ways and was often based on definitions that were inconsistent. Without consistent practices for measurement, no clear, quantitative picture of software development capabilities was possible, nor was there any way to compare products and processes. Therefore, a standard and reliable set of measures that would help acquisition program managers and software development contractors alike was an early priority for the SEI. In response to the DoD's 1991 Software Technology Strategy, the SEI agreed to lead the development of a set of core measures to "help the DoD plan, monitor, and manage its internal and contracted software development projects" [Carleton 1992]. The resulting definition frameworks made it possible for organizations to use measures that best matched their processes and infrastructure while benefiting from a standard way of describing the operational definitions in detail.

Once the definition frameworks were developed, the SEI began work on the problem of helping organizations decide what was useful to measure, having realized that many reports could be generated that had little to no impact on decision making within the organization. The SEI began to investigate the goal-question-metric method for aligning measurement with information needs in the organization [Basili 1984]. The SEI modified the approach to include explicit consideration of the indicator (or output) of the measurement activity that would be used by decision makers, thus creating the goal-question-*indicator*-measure (GQ(I)M) method.

Substantial work followed on the application of Six Sigma analytical techniques to software engineering [Penn 2007]. The Six Sigma connection provided a rich set of tools as well as a "brand" that already had roots in many organizations, which facilitated its adoption. The work has further evolved to incorporate techniques used in Six Sigma to help improve estimates developed early in the DoD acquisition lifecycle.

The SEI's management work has had a global impact: use of its models and processes has been documented in 74 countries across the globe and on every continent except Antarctica. Further, SEI software engineering process techniques allow companies to outpace typical industry performance in terms of the quality and predictable delivery of the software they develop. Productivity improvements of 25 percent or more have been demonstrated, along with testing cost and schedule reductions of up to 80 percent, and cost savings of about 25-50 percent per software product release. The 2009 book *Software Engineering Best Practices: Lessons from Successful Projects in the Top Companies* [Jones 2009] ranked TSP and CMMI as "top software engineering practices" in terms of enabling superior software development performance. Beyond improving development performance, the SEI's management work has provided the framework for increased operational resilience and better management of risks. Those working on complex systems use SEI management techniques to create more accurate and precise cost estimations, improve insight into the causes of failure, increase reliability for fielded systems with lower total cost of ownership, and understand early warning signs of cost, schedule, or quality concerns.

### 4.0.13 References

[Averill 1993] Averill, Edward; Byrnes, Paul; Dedolph, Michael; Maphis, John; Mead, Maphis; & Puranik, Rajesh. *Software Capability Evaluation (SCE) Version 1.5 Method Description* (CMU/SEI-93-TR-017). Software Engineering Institute, Carnegie Mellon University, 1993. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=11921

[Barbacci 1985] Barbacci, M. R.; Habermann, A. N.; & Shaw, M. "The Software Engineering Institute: Bridging Practice and Potential." *IEEE Software 2,* 6 (November 1985): 4-21.

[Basili 1984] Basili, V. & Weiss, D. A. "Methodology for Collecting Valid Software Engineering Data." *IEEE Transactions on Software Engineering 10,* 3 (November 1984): 728-738.

[Bernard 2005] Bernard, Thomas; Gallagher, Brian; Bate, Roger; & Wilson, Hal. *CMMI Acquisition Module (CMMI-AM), Version 1.1* (CMU/SEI-2005-TR-011). Software Engineering Institute, Carnegie Mellon University, 2005. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=7665

[Boehm 1981] Boehm, Barry. *Software Engineering Economics*. Prentice-Hall, 1981 (ISBN 0138221227).

[Caralli 2010] Caralli, Richard; Allen, Julia; Curtis, Pamela; White, David; & Young, Lisa. *CERT Resilience Management Model, Version 1.0* (CMU/SEI-2010-TR-012). Software Engineering Institute, Carnegie Mellon University, 2010. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9479

[Carleton 1992] Carleton, Anita; Park, Robert; Bailey, Elizabeth; Goethert, Wolfhart; Florac, William; & Pfleeger, Shari. *Software Measurement for DoD Systems: Recommendations for Initial Core Measures* (CMU/SEI-92-TR-019). Software Engineering Institute, Carnegie Mellon University, 1992. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=11675

[CMMI 2010a] CMMI Product Team, *CMMI for Development, Version 1.3* (CMU/SEI-2010-TR-033). Software Engineering Institute, Carnegie Mellon University, 2010. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9661

[CMMI 2010b] CMMI Product Team, *CMMI for Services, Version 1.3* (CMU/SEI-2010-TR-034). Software Engineering Institute, Carnegie Mellon University, 2010. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9665

[Curtis 2003] Curtis, William; Hefley, William; & Miller, Sally. *People Capability Maturity Model (P-CMM), Version 2.0, Second Edition* (CMU/SEI-2009-TR-003). Software Engineering Institute, Carnegie Mellon University, 2009. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9071

[Davenport 2005] Davenport, Thomas. "The Coming Commoditization of Processes." *Harvard Business Review 83,* 6 (June 2005)*:* 101-108.

[Davis 2003] Davis, Noopur & Mullaney, Julia. *The Team Software Process (TSP) in Practice: A Summary of Recent Results* (CMU/SEI-2003-TR-014). Software Engineering Institute, Carnegie Mellon University, 2003. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=6675

[DoD 1982] Department of Defense. *Report of the DoD Taskforce on Software Problems* (Stock No. ADA123449). National Technical Information Service, 1982.www.dtic.mil/docs/citations/ADA123449

[Druffel 1983] Druffel, Larry E.; Redwine, Samuel T. Jr.; & Riddle, William E. "The STARS Program: Overview and Rationale." *IEEE Computer 16,* 11 (November 1983): 21-29.

[Fowler 1990] Fowler, P. & Rifkin, S. *Software Engineering Process Group Guide* (CMU/SEI-90-TR-24.) Software Engineering Institute, Carnegie Mellon University, 1990. http://resources.sei.cmu.edu/asset_files/TechnicalReport/1990_005_001_15881.pdf

[Herbsleb 1997] Herbsleb, James; Zubrow, David; Goldenson, Dennis; Hayes, Will; & Paulk, Mark. "Software Quality and the Capability Maturity Model." *Communications of the ACM 40,* 6 (June 1997): 30-40.

[Humphrey 1985] Humphrey, W. S. "The IBM Large-Systems Software Development Process: Objectives and Direction." *Large-System Software Development 24*, 2 (1985): 76.

[Humphrey 1988] Humphrey, W. S. *A Discipline for Software Engineering*. Addison-Wesley, 1988 (ISBN:0201546108).

[Humphrey 1989] Humphrey, Watts S. *Managing the Software Process.* Addison-Wesley Professional, 1989 (ISBN: 0201180952). http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=30884

[Humphrey 2000] Humphrey, Watts. *The Team Software Process (TSP)* (CMU/SEI-2000-TR-023). Software Engineering Institute, Carnegie Mellon University, 2000. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=5287

[Jones 2009] Jones, Capers. *Software Engineering Best Practices: Lessons from Successful Projects in the Top Companies.* McGraw-Hill, 2009 (ISBN 007162161X).

**[**Kitson 1993**]** Kitson, David H. & Masters, Stephen M. "An Analysis of SEI Software Process Assessment Results 1987-1991." *Proceedings of the 15th International Conference on Software Engineering (ICSE)*. Baltimore, Maryland, May 17-21, 1993. IEEE Computer Society/ACM Press, 1993

[Paulk 1993] Paulk, Mark; Curtis, William; Chrissis, Mary Beth; & Weber, Charles. *Capability Maturity Model for Software (Version 1.1)* (CMU/SEI-93-TR-024). Software Engineering Institute, Carnegie Mellon University, 1993. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=11955

[Penn 2007] Penn, M. L.; Siviy, Jeannine; & Stoddard, Robert W. *CMMI and Six Sigma: Partners in Process Improvement.* Addison-Wesley Professional, 2007. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=30452

[SEI 1995] *A Systems Engineering Capability Maturity Model, Version 1.1* (CMU/SEI-95-MM-003). Software Engineering Institute, Carnegie Mellon University, 1995. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=12291

[SGMM 2011] SGMM Team, *Smart Grid Maturity Model, Version 1.2: Model Definition* (CMU/SEI-2011-TR-025). Software Engineering Institute, Carnegie Mellon University, 2011. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=10035

[Zubrow 1994] Zubrow, David; Hayes, William; Siegel, Jane; & Goldenson, Dennis. *Maturity Questionnaire* (CMU/SEI-94-SR-007). Software Engineering Institute, Carnegie Mellon University, 1994. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=12099

[Zubrow 2003] Zubrow, David, *CMMI Adoption Trends,* 2003. http://www.sei.cmu.edu/library/abstracts/news-at-sei/feature14q03.cfm

## 4.1 The Capability Maturity Model for Software

### 4.1.1 The Challenge: Consistent and Predictable Management of Software Development.

In 1986, there was a general realization in the DoD and among defense contractors that certain software engineering practices produced working software with increased consistency. However, relatively few practitioners recognized these largely undocumented practices. Most companies had established their own practices through experience, but the importance that software project planning and tracking, commitment management, quality management, and configuration management practices have in successfully managing software development—and the need for organizational support to perform these practices effectively—was not broadly appreciated. Furthermore, software tools supporting such activities were often developed by the organization and displayed the idiosyncrasies particular to that organization. Frequently, the tool choice dictated the defining and interaction of the processes. Nevertheless, examples of successful software development programs did occur and the recognition that software could be managed by a defined and measured process was reflected in an expanding body of literature by leading authors.

The DoD identified the management of the software component of programs as a major problem area [DoD 1982], and the DoD STARS strategy envisioned a managed process well supported by automated tools [Druffel 1983].

### 4.1.2 A Solution: The Capability Maturity Model for Software

In response to the DoD need, the SEI strategic plan identified the management of the software development process—both by DoD program offices and defense contractors—as a fundamental focus of the SEI [Barbacci 1985]. The SEI recruited a retiring executive from IBM who had been instrumental in creating and managing IBM efforts toward greater software quality and predictability and began work to define a process management framework in 1986. Shortly thereafter, the Air Force program manager asked the SEI to conduct a study of “best practices.” The study became key to the SEI’s efforts to define and implement its process management framework. Several workshops were conducted with leading software professionals in the DoD, defense industry, commercial industry, and academia to develop a consensus on the practices that consistently lead to improved software development. To help organizations determine how well their work stacked up against these practices, the SEI produced a Maturity Questionnaire [Humphrey 1988]. Response to this questionnaire was overwhelmingly positive, both from the DoD and the defense industry.

Initially, the questionnaire identified a five-level model of organizational maturity based on the implementation of software process management principles [Humphrey 1989]. After assisting several organizations with their assessments and subsequent improvement efforts, the SEI produced a guide for how organizations might manage that process [Fowler 1990]. As the community began to adopt these ideas, they expressed a need for a more precise definition of the practices and the underlying model.

The SEI developed a more explicit model with practices focused on establishing a richer set of constructs in which to place and organize the practices within each maturity level. One such construct was the "key process area" with its goals and key practices (another was the "common features"). The key process areas were identified through three sources: (1) problem areas and practices associated with each maturity level [Humphrey 1989], (2) multiple practitioner community reviews and workshops, and (3) statistical analyses of both assessment and questionnaire data (e.g., could the results of the latter predict the former?). Key process areas also became a mechanism for reporting and rating the results of an assessment and setting targets for process improvement and would serve as a basis for developing and updating the questionnaire, which would now serve more of a diagnostic purpose.

As a result of these activities, the SEI published the Software Capability Maturity Model [Paulk 1993] and updated the software process assessment and software capability evaluation methods to make use of the explicit model.

### 4.1.3 The Consequence: A Revolutionary International Movement

With the publication of a practitioner community-vetted explicit model of software development practice, worldwide adoption grew each year for many years. The SEI created user group meetings, later called Software Engineering Process Group (SEPG) conferences, as a means of interacting with the practitioner community and broadly sharing lessons learned. Software Process Improvement Networks (SPINs), organizations or individuals regularly hosting meetings for co-located software process improvement champions and the curious, grew in number from two in 1992 to about fifty by 1998. The number of formal assessments also grew dramatically.

The Software CMM thus launched the process improvement movement, and other CMMs emerged [Bate 1994, Konrad 1996] as other communities recognized their value [Davenport 2005].

**The View from Others**

*The model, created in 1987, has become a worldwide standard for software development processes and is now embedded within many government and industry organizations. It has provided an objective basis for measuring progress in software engineering and for comparing one software provider's processes to another's. This in turn has facilitated the growth of offshore providers in India and China by commoditizing software development processes and making them more transparent to buyers.*

– Thomas Davenport, currently President's Distinguished Professor in Management and Information Technology at Babson College [Davenport 2005]

*The CMM is not a panacea and it does not solve all problems. In fact, a case could be made that the CMM creates a few problems of its own. In general, however, the superimposition of the CMM structure on a good sized organization has benefited it wherever that has occurred.*

– Capers Jones, Chief Scientist Emeritus, Software Productivity Research [CAI 2005]

### 4.1.4 The SEI Contribution

The ideas in the CMM for Software were contributed by many people. Many of those ideas preceded the SEI effort, such as those from authors identified in the footnote, and from the IBM and Texas Instruments experiences. In addition, as the SEI began to lead the effort, many software engineers contributed from their own experiences. The SEI contribution was to provide leadership to the community, assimilating and filtering the ideas into a consistent framework to produce the documents that became a worldwide de facto standard for software process improvement. The SEI designed a new structure for visualizing the evolution of practices, a seminal information architecture that was mimicked and adapted over time. The process of providing leadership to the community in a consensus-building effort has become the hallmark of SEI efforts.

The Software CMM eventually became superseded by CMMI, especially CMMI for Development and CMMI for Acquisition. The SEI undertook initiatives to unite two disciplines and communities that sometimes acted at odds with each other and yet must work together to ensure product and program success: namely, systems engineering and software engineering

### 4.1.5 References

[Barbacci 1985] Barbacci, M. R.; Habermann, A. N.; & Shaw, M. "The Software Engineering Institute: Bridging Practice and Potential." *IEEE Software 2*, 6 (November 1985): 4-21 (ISSN: 0740-7459).

[Bate 1994] Bate, Roger; Reichner, Albert; Garcia-Miller, Suzanne; Armitage, James; Cusick, Kerinia; Jones, Robert; Kuhn, Dorothy; Minnich, Ilene; Pierson, Hal; & Powell, Tim. *A Systems Engineering Capability Maturity Model, Version 1.0* (CMU/SEI-94-HB-004). Software Engineering Institute, Carnegie Mellon University, 1994. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=12037

[CAI 2005] Computer Aid Inc. "Focus on Capers Jones, Chief Scientist Emeritus, SPR, A CAI State of the Practice Interview." July 2005. http://www.compaid.com/caiinternet/casestudies/capersjonesinterview1.pdf

[Davenport 2005] Davenport, Thomas. "The Coming Commoditization of Processes." *Harvard Business Review.* (June 2005): 101-108.

[DoD 1982] Department of Defense. *Report of the DoD Taskforce on Software Problems* (Stock No. ADA123449). National Technical Information Service, 1982.www.dtic.mil/docs/citations/ADA123449

[Druffel 1983] Druffel, L. E.; Redwine, Jr., S. T.; Riddle, W. E. "The STARS Program: Overview and Rationale." *Computer 16*, 11 (November 1983): 21-29.

[Fowler 1990] Fowler, Priscilla & Rifkin, Stanley. *Software Engineering Process Group Guide* (CMU/SEI-90-TR-024). Software Engineering Institute, Carnegie Mellon University, 1990. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=11253

[Humphrey 1988] Humphrey, Watts; Sweet, William; Edwards, R.; LaCroix, G.; Owens, M.; & Schulz, H. A *Method for Assessing the Software Engineering Capability of Contractors* (CMU/SEI-87-TR-023). Software Engineering Institute, Carnegie Mellon University, 1988. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=10345

[Humphrey 1989] Humphrey, Watts S. *Managing the Software Process.* Addison-Wesley Professional, 1989 (ISBN: 0201180952).

[Konrad 1996] Konrad, M.; Chrissis, M. B.; Ferguson, J.; Garcia, S.; Hefley, B.; Kitson, D.; & Paulk, M. "Capability Maturity Modeling at the SEI." *Software Process: Improvement and Practice 2*, 1 (March 1996): 21-34.

[Paulk 1993] Paulk, Mark; Curtis, William; Chrissis, Mary Beth; & Weber, Charles. *Capability Maturity Model for Software (Version 1.1)* (CMU/SEI-93-TR-024). Software Engineering Institute, Carnegie Mellon University, 1993. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=11955

## 4.2 Appraisal Methods

### 4.2.1 The Challenge: Predicting Software Engineering Performance

As the importance of software grew substantially in DoD procurements, so did the need for evaluating software contractors' abilities to competently perform on software engineering contracts. Meanwhile, a team at IBM was also investigating its own software engineering performance and noticed that different IBM sites varied in their levels of predictable performance. To find out why, people began to collect the factors contributing to success and compare sites on this basis, which enabled them to identify issues to tackle and best practices to emulate. A structured approach began to emerge from this work that allowed for site visits for gathering relevant information, identifying opportunities, and establishing priorities for improvement. As problems with software contractors came to a head in the mid-1980s, the DoD turned to the SEI.

### 4.2.2 A Solution: Assessing the Capability of Contractors

The DoD asked the SEI to use its "objective broker" status to help figure out how to fairly and effectively determine a contractor's likely performance based on that contractor's capability when producing software. In response, the SEI began to increase its role in software process assessments, drawing on the work at IBM and other organizations.

In 1986, the SEI work had begun in earnest towards establishing a formal process for evaluating potential software performance, when the U.S. Air Force and MITRE Corp. asked it to develop a site assessment method and related model that could be used to assess commercial and government software organizations. SEI experts began concentrating on a process that would facilitate objective and consistent assessments of the ability of potential contractors to develop software in accordance with up-to-date software engineering methods.

In 1987, *A Method for Assessing the Software Engineering Capability of Contractors* was published [Humphrey 1988]. The primary goal was to provide a standardized, publicly available method that could be periodically reviewed and modified. The method was a structured assessment approach intended to augment contractor evaluation methods in use at the time. The method document included a Maturity Questionnaire, a five-level Process Maturity Framework, and a brief set of guidelines for conducting an assessment and evaluating the results.

Even before the Maturity Questionnaire was formally published, the SEI technical staff recognized the need for a more detailed description of the assessment process and, in July 1987, published a preliminary report on conducting assessments [Humphrey 1987]. It is interesting to note that this report envisioned internal assessments for process improvement and evaluation assessments conducted by DoD procurement personnel. This integration of process assessment methods soon dissolved, resulting in two similar but separate methods and constituencies: Software Process Assessment and Software Capability Evaluation. Each of the method descriptions had requirements for team selection and training, planning, conducting on-site activities, and reporting results.

Software Process Assessments (SPAs), originally called SEI-Assisted Assessments, were first described in 1989. The SPA was used to identify an organization's major problems and engage

opinion leaders and senior managers in the change process. The goal was to prioritize areas for improvement and to provide guidance on how to make those improvements. This more structured method was originally performed exclusively by SEI staff. Later, "SPA Vendors" were trained to perform the assessments; this small community of vendors later grew into a worldwide program of licensed assessors.

Software Capability Evaluations (SCEs) were conducted during the same time period, but with different goals in mind. They were used primarily to evaluate sources and contractors by gaining insight into the software process capability of a supplier organization and were intended to contribute to better acquisition decisions, improve subcontractor performance, and provide insight for a purchasing organization. SCEs were used in software acquisition as a discriminator to select suppliers, for contract monitoring, and for awarding incentives [SCE 1993].

SPAs and SCEs sometimes produced inconsistent results, which led the SEI to develop the CMM Appraisal Framework (CAF) as an appraisal method standard to address the issue [Masters 1995]. With the publication of the Capability Maturity Model (CMM) for Software, the SEI released CAF-compliant methods for assessment (CBA IPI [Dunaway 1996]) and evaluation (SCE V3.0 [Byrnes 1996]).

At the turn of the millennium, representatives in government and industry asked for the development of an integrated model that would improve the usability of maturity models by integrating many different models into one framework. The Capability Maturity Model Integration (CMMI) team published the appraisal requirements for CMMI (ARC), ushering in a new era for appraisals [CMMI 2001]. The Standard CMMI Appraisal Method for Process Improvement (SCAMPI) was developed [AMIT 2001], along with the specification for two other appraisal classes. Later the SEI developed SCAMPI B and C as a 100 percent community-funded project [Hayes 2005]. Factors that might influence an organization's choice of a SCAMPI (A, B, or C) include cost, schedule, accuracy, efficiency, and the desired results. SCAMPI continues to have a wide range of uses, including internal process improvement and external capability determinations.

### 4.2.3 The Consequence: Reduced Risk in Selecting Contractors

The SEI work on assessing/evaluating contractors led the DoD and other government acquisition organizations to change their criteria for selecting contractors. They now consider the discipline in the contractors' software development and how well they follow their defined processes. As a result, the DoD and others have reduced their risk in acquiring software.

Having a published, consistent method enables contractors to prepare in advance for an assessment, and the feedback from assessments helps organizations identify areas where they should improve their own capabilities. The objective evidence from the assessments helps to build their organization's commitment to improvement and allows comparisons of results with those of other organizations. Trends and results can be tracked through the process Maturity Profile, where appraisal results are aggregated and published periodically. The SEI published benchmarks that provided in-depth analysis of data and trends from organizations participating in assessments [Kitson 1993, Hayes 1995]. Additionally, the SEI found that assessments were viewed to be accurate and useful in guiding subsequent process improvement efforts [Herbsleb 1997].

Ultimately, the SEI's work in establishing process appraisal methods has allowed the armed services to improve their ability to serve the national interest by awarding contracts to the software contractors with the best capability.

### 4.2.4 The SEI Contribution

The appraisal methods published by the SEI were all developed in partnership with government and industry. The formality of the appraisal methods—and the models on which they are based—led to repeatable, understandable results that have helped industry and government recognize the value of process improvement. As a result, organizations have been able to increase the consistency of their software development performance and, presumably, the quality of their software.

The SEI program in which "SPA Vendors" were trained to perform Software Process Assessments was first limited to a small community of vendors. This program later grew into a worldwide program as the SEI assessment methods evolved, along with the SEI licensing paradigm. This SEI activity thus spawned a new industry that still flourishes today.

The commercialization of process improvement using primarily CMM and CMMI-based appraisal methods has helped to highlight one of the many contributions of the SEI and Carnegie Mellon to the software engineering community. CMM/CMMI maturity levels have become a de facto international standard. The SEI's contribution includes creating the profession of SCAMPI Lead Appraisers/process improvement professionals. This was done through certification (500 as of 2013) and based on the SCAMPI Lead Appraiser Body of Knowledge (SLA BOK) [Masters 2007].

### 4.2.5 References

[AMIT 2001] Members of the Assessment Method Integrated Team. *Standard CMMI Appraisal Method for Process Improvement (SCAMPI), Version 1.1: Method Definition Document* (CMU/SEI-2001-HB-001). Software Engineering Institute, Carnegie Mellon University, 2001. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=5325

[Byrnes 1996] Byrnes, Paul & Phillips, Michael. *Software Capability Evaluation Version 3.0 Method Description* (CMU/SEI-96-TR-002). Software Engineering Institute, Carnegie Mellon University, 1996. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=12503

[CMMI 2001] CMMI Product Team. *Appraisal Requirements for CMMI, Version 1.1 (ARC, V1.1)* (CMU/SEI-2001-TR-034). Software Engineering Institute, Carnegie Mellon University, 2001. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=5781

[Dunaway 1996] Dunaway, Donna & Masters, Steve. *CMM-Based Appraisal for Internal Process Improvement (CBA IPI): Method Description* (CMU/SEI-96-TR-007). Software Engineering Institute, Carnegie Mellon University, 1996. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=12533

[Hayes 1995] Hayes, William & Zubrow, David. *Moving On Up: Data and Experience Doing CMM-Based Process Improvement* (CMU/SEI-95-TR-008). Software Engineering Institute, Carnegie Mellon University, 1995. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=12353

[Hayes 2005] Hayes, William; Miluk, Gene; Ming, Lisa; Glover, Margaret; & Members of the SCAMPI B and C Project. *Handbook for Conducting Standard CMMI Appraisal Method for Process Improvement (SCAMPI) B and C Appraisals, Version 1.1* (CMU/SEI-2005-HB-005). Software Engineering Institute, Carnegie Mellon University, 2005. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=7281

[Herbsleb 1997] Herbsleb, James; Zubrow, David; Goldenson, Dennis; Hayes, Will; & Paulk, Mark. "Software Quality and the Capability Maturity Model." *Communications of the ACM 40*, 6 (June 1997): 30-40.

[Humphrey 1987] Humphrey, Watts & Kitson, David. *Preliminary Report on Conducting SEI-Assisted Assessments of Software Engineering* (CMU/SEI-87-TR-016). Software Engineering Institute, Carnegie Mellon University, 1987. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=10307

[Humphrey 1988] Humphrey, Watts; Sweet, William; Edwards, R.; LaCroix, G.; Owens, M.; & Schulz, H. A. *Method for Assessing the Software Engineering Capability of Contractors* (CMU/SEI-87-TR-023). Software Engineering Institute, Carnegie Mellon University, 1988. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=10345

[Kitson 1993] Kitson, David H. & Masters, Stephen M. "An Analysis of SEI Software Process Assessment Results 1987-1991." *Proceedings of the 15th International Conference on Software Engineering*. ICSE, Baltimore, Maryland, May 17-21, 1993. IEEE Computer Society/ACM Press, 1993.

[Masters 1995] Masters, Steve & Bothwell, Carol. *CMM Appraisal Framework, Version 1.0* (CMU/SEI-95-TR-001). Software Engineering Institute, Carnegie Mellon University, 1995. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=12323

[Masters 2007] Masters, Steve; Behrens, Sandra; Mogilensky, Judah; & Ryan, Charles. *SCAMPI Lead Appraiser Body of Knowledge (SLA BOK)* (CMU/SEI-2007-TR-019). Software Engineering Institute, Carnegie Mellon University, 2007. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=8455

[SCE 1993] Software Capability Project. *Software Capability Evaluation (SCE) Version 1.0 Implementation Guide* (CMU/SEI-93-TR-018). Software Engineering Institute, Carnegie Mellon University, 1993. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=11935

## 4.3 Maturity Profile

### 4.3.1 The Problem: Lack of Data on Use of SEI Models and Appraisal Results

Although the SEI had developed a model for assessing contracting risk, leading to the CMM for Software and CMMI, there initially was no data available on the adoption of the models or the results of the appraisals based on the models.

### 4.3.2 A Solution: Community Maturity Profile

The initial approach to addressing the problem was publication of a report on the first 10 assessments [Humphrey1989]. A second report summarized the history of results from 59 assessments conducted from 1987 through 1991 [Kitson 1992]. While these reports were generally found to be helpful, the SEI realized the community wanted more information about the process maturity of the software engineering community—a maturity profile.

To satisfy this need for improved information, the SEI needed to take a number of steps. First, the reports needed to be produced as a matter of routine, not as an afterthought. Second, there needed to be some standardization of the results and contextual information collected and reported. Third, the SEI needed to develop the processes, procedures, and infrastructure for having results reported to it by those performing assessments. Finally, the SEI needed to develop the method for analyzing the assessment data.

To encourage the reporting of assessment results, the SEI established a routine schedule, in 1992, for producing a summary and analysis of the accumulated assessment results. Providing updated information of value back to the community motivated those conducting assessments to turn in their results to the SEI. As more results were turned in, more analyses could be conducted. The ideas for those analyses came from members of the community as well as SEI researchers. As the volume of data started to grow and the use of the Maturity Profile grew, it became clear that the reporting needed to be standardized and additional information needed to be collected to support the desired analyses. The new data included characteristics of the organization, the team conducting the assessment, the assessment itself, and greater detail about the results.

The SEI created a database and a series of internal reports and processes for managing the data, reporting assessment results, and interacting with those reporting results. Creating the database was a major step toward enabling the routine reporting of results and tracking of trends in organizations and the community over time. Furthermore, tracking trends in organizations required a set of business rules for persistent identification in light of mergers and acquisitions. The SEI also developed a set of processes and procedures for tracking the conduct and reporting of assessments. This initially started as a reconciliation process where those reporting assessments and the SEI would compare their records. Eventually, the SEI required registration of the appraisal plans in advance of conducting the appraisal. The appraisal results were then required to be reported within 30 days.

The content of the Maturity Profile evolved over the years. The primary profile was the simple display of the percentage of organizations at each maturity level. Over time, an increasing number of data subsets for maturity profiles were reported. Also tracked and reported was the time to

move from maturity level to maturity level, as well as those regressing, that is, those assessed at a lower maturity level.

The Community Maturity Profile helped to spur a closer connection between the SEI and its partners who performed assessments. The summarization and routine reporting of updated information was seen as a valuable resource that the SEI provided back to its partners and the community. As a result, the partners became more diligent about reporting assessment information to the SEI. Similarly, the SEI took its obligation to produce the profile on a semi-annual basis seriously.

### 4.3.3 The Consequence: Reliable Source of Data for the Community

The Community Maturity Profile became the principal source of information regarding the adoption of CMM and CMMI and the state of the community in terms of process improvement [Zubrow 2003]. The profile and its data have been used in a myriad of organizational settings to benchmark current process maturity and to develop business cases for process improvement. It has also been used as a basis for academic research and incorporated into courses on software engineering and process improvement.

The publication of this valuable information is unique among process improvement methods. There are no other methods with a similar depth and breadth of information regarding their adoption and results. The information in the profile has been incorporated into many publications and presentations as the authoritative source on the status of organizations engaged in CMM/CMMI-based process improvement as well as trends in the community, such as how long it takes to move from one maturity level to the next.

### 4.3.4 The SEI Contribution

The SEI developed the processes, procedures, and a database that enabled it to publish the Community Maturity Profile as a routine report summarizing data reported from organizations that have conducted CMMI-based appraisals. The process for collecting the data and the profile itself evolved over the years to provide a better and more complete depiction of the status and trends stemming from the of CMM/CMMI. The Community Maturity Profile is the primary source of information within the software process improvement community regarding the adoption and use of CMM/CMMI.

### 4.3.5 References

[Humphrey 1989] Humphrey, Watts; Kitson, David; & Kasse, Timothy. *The State of Software Engineering Practice: A Preliminary Report* (CMU/SEI-89-TR-001). Software Engineering Institute, Carnegie Mellon University, 1989. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=10851

[Kitson 1992] Kitson, David & Masters, Steve. *An Analysis of SEI Software Process Assessment Results 1987-1991* (CMU/SEI-92-TR-024). Software Engineering Institute, Carnegie Mellon University, 1992. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=11709

[Zubrow 2003] Zubrow, David. “CMMI Adoption Trends.” news@sei. Software Engineering Institute, Carnegie Mellon University, 2003. http://www.sei.cmu.edu/library/abstracts/news-at-sei/feature14q03.cfm

## 4.4 The People Capability Maturity Model

### 4.4.1 The Challenge: Assessing and Improving Workforce Capability

Following successful implementation of the CMM for Software, organizations discovered that they could quantify how well they developed their products, but could not tell if they were utilizing and deploying their employees to maximum efficiency and uniform productivity. Organizational competency and capability were driving contractual commitments and return on investment, and organizations asked for a model similar to the CMM to assess and improve organizational competency in the management of its human resources. The U.S. Army, in particular, encouraged the SEI to address this need and provided initial funding.

### 4.4.2 A Solution: The People CMM

The SEI responded to this need by developing the People Capability Maturity Model. Since the demand for the People CMM and its companion product suite was driven by the success of the Software CMM, the People CMM had its roots firmly planted in the SEI process maturity framework. In addition, the SEI included best practices in human capital management and the measurement of organizational change.

In 1998, the People CMM and its companion courses and appraisals were released after three years of development and rigorous review [Hefley 1998]. While initial funding to develop and test the People CMM was provided by the DoD, and early appraisals conducted with the U.S. Army, later support and funding was primarily provided by organizations providing products and services not only to the DoD, but to other agencies within the U.S. government, such as the Federal Emergency Management Agency (FEMA). Other examples of funders are GDE Systems, Boeing, BAE Systems, Lockheed Martin, and Computer Sciences Corp.

When improvements guided by the People CMM are initiated, they are sometimes perceived as a human resources program. However, organizations have uniformly found the People CMM to be a general business excellence model. The increasing focus on performance

**The View from Others**

*Intel Information Technology (Intel IT) supports the computing needs of over 80,000 employees in more than 70 sites worldwide. Intel IT sources, designs, develops, implements and maintains the hardware, software and IT solutions that enable the company to operate efficiently. (page 132)*

*Intel IT decided that the People CMM was the most appropriate model for attaining its objectives of developing a world-class workforce and organization capabilities for IT by strategically shaping its future workforce and influencing its partners and industry. (page 133)*

*Over the course of three years, Intel IT achieved many of its innovation goals, including a 200 percent increase in patents emerging from the IT workforce, and solid improvements in employee feedback about the organization's leadership and Great Place to Work scores. (page 136)*

– Jack Anderson, Chair, Innovation Management Working Group, Innovation Value Institute (a consortium founded by Intel of over 35 companies that have come together to improve the IT industry) [Curtis 2009] case study

improvement causes People CMM results to affect operational performance of units and the organization as a whole. The People CMM provides guidance that improves an organization's ability to satisfy the identified business objectives by deploying a competent, capable workforce that is executing and continuously improving its business processes.

In 2001, the People CMM was updated using lessons learned, change requests from users, input from the SEI advisory board, and the latest research on organizational quality improvement. The book, The *People Capability Maturity Model, Guidelines for Improving the Workforce*, was published as part of the Addison-Wesley SEI Series [Hefley 2002]. It contains new guidance for users as well as case studies from organizations using the model. The case studies identify a trend of use in organizations providing products and services to the public; case studies include Novo Nordisk IT/AS, Europe and Tata Consulting Services, India.

A second edition of the 2002 publication of the People CMM was released in 2010 [Curtis 2009]. This updated version has made no change to the model but has added more examples and explanatory subpractices. In addition, the front matter has been expanded to provide more guidance to the reader and user, and new case studies added, some of which were written entirely by People CMM customers. Newly titled *People CMM, Second Edition, A Framework for Human Capital Management*, the book focuses on how the People CMM is being applied and by what types of organizations. It contains seven new case studies: Boeing, Intel Information Technology, Pfizer Worldwide Technology, Ericsson, Accenture, Club Mahindra, HCLT BPO, and Tata Consultancy Services. While case studies provide an insight into the use of the People CMM as a guide and demonstrate an organization's journey as they mature, it must be noted that some specifics have been omitted as they are considered competitive advantage.

### 4.4.3 The Consequence: A Competent Workforce That Can Meet Business Goals

The People CMM clearly meets a perceived as well as stated need. Organizations have guidance enabling them to improve their ability to satisfy their identified business objectives by deploying a competent, capable workforce that is executing and continuously improving its business processes.

In addition to DoD organizations, defense contractors, and other government agencies, national and even international organizations have adopted the model as the basis for their improvement efforts. Such organizations include the areas of business process outsourcing, hospitality, construction, insurance, energy and utilities, banking and financial services, information technology, consulting, pharmaceutical, software development, and management information systems.

The People CMM has also been used to support and sustain the attainment of CMMI maturity levels by building competencies and a workforce that can successfully execute and manage organizational business processes. Use of the People CMM has now been verified in Europe, Asia, and Australia as well as in North America.

### 4.4.4 The SEI Contribution

As is the case with many other SEI efforts and all of the Capability Maturity Model efforts, the SEI provided the vision and leadership for the creation of the People CMM and managed the process. But the content was a community effort, with many experts from the international community offering their perspectives during the initial development. The subsequent improvements in 2001 and 2010 were driven largely by the experiences of organizations that used the model and its supporting materials.

### 4.4.5 References

[Curtis 2009] Curtis, Bill; Hefley, William E.; & Miller, Sally A. *People CMM: A Framework for Human Capital Management*, 2nd ed. Addison-Wesley Professional, 2010 (ISBN 032155390X).

[Hefley 1998] Hefley, William & Curtis, William. *People CMM-Based Assessment Method Description* (CMU/SEI-98-TR-012). Software Engineering Institute, Carnegie Mellon University, 1998. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=13125

[Hefley 2002] Hefley, William E.; Miller, Sally A; & Curtis, Bill (2002). *The People Capability Maturity Model: Guidelines for Improving the Workforce*. Addison-Wesley Professional, 2002 (ISBN 0-201-60445-0).

## 4.5 Managing Operational Resilience

### 4.5.1 The Challenge: Delivering Essential Services in the Presence of Stress and Disruption

Beginning in the late 1990s, the DoD faced a set of problems shared with organizations in every sector—U.S. federal government agencies, defense and commercial industry, and academia—arising from increasingly complex business and operational environments. Most organizations continue to be constantly bombarded with conditions and events that introduce stress and uncertainty that may disrupt effective operation. Stress related to operational resilience—the ability of an organization to achieve its mission even under degraded circumstances—can come from many sources, including risks and threats resulting from technology advances and the increasing globalization of organizations and their supply chains.

All these demands conspire to force organizations to rethink how they perform operational risk management and how they address the resilience of high-value business services and processes. The traditional, and typically compartmentalized, disciplines of security, operational continuity, and information technology (IT) operations must be expanded to provide protection and continuity strategies for high-value services and supporting assets that are commensurate with these new operating complexities.

**The View from Others**

*Our comprehensive analysis of business resilience management models identified CERT-RMM as the most promising model for use within our enterprise due to promoting convergence, modeling the needs of a large enterprise, considering risks for both protecting and sustaining assets, and its focus on measuring and institutionalizing resilience processes.*

– Nader Mehravari, former director of Corporate Business Resiliency Strategic Initiative, Lockheed Martin Corp. [Caralli 2010, Ch. 7]

*CERT-RMM helps us define the processes by which we conduct incident responses for security incidents, including how we interact with the other business units and the CISO's [chief information security officer's] office for the recovery of evidence and continuity of operations.* [Joch 2013]

### 4.5.2 A Solution: Convergence of Operational Risk Disciplines That Accelerated the SEI's Ability to Tackle Resilience

In 1999, the SEI released the Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE) method for information security risk management. OCTAVE provided a new way to look at information security risk from an operational perspective and asserted that operational (business) people are in the best position to identify and analyze security risk. This effectively repositioned IT's role in security risk assessment and placed the responsibility closer to the operations activity in the organization [Alberts 1999].

In October 2003, a group of 20 IT and security professionals from defense organizations, the financial services sector, IT, and security services met at the SEI to begin building an executive-level community of practice for IT operations and security. The desired outcome was to better capture and articulate the relevant bodies of knowledge that enable and accelerate IT operational

and security process improvement. The bodies of knowledge identified included IT and information security governance, audit, risk management, IT operations, security, project management, and process management.

In December 2004, the SEI released a technical note titled *Managing for Enterprise Security* [Caralli 2004] that introduced operational resilience as the objective of security activities and began to describe the convergence between security management, business continuity management, and IT operations management as essential for managing operational risk. In March 2005, the SEI hosted a meeting with representatives of the Financial Services Technology Consortium (FSTC).[16] The FSTC's Business Continuity Standing Committee was actively organizing a project to explore the development of a reference model to help determine an organization's capability to manage operational resilience as a follow-on to lessons learned in the aftermath of Sept. 11, 2001. The respective efforts were clearly focused on solving the same problem: How can an organization predictably and systematically control operational resilience through activities such as security and business continuity?

**The View from Others**

*The CERT-RMM class provided Lockheed Martin participants with a solid framework for measuring organizational and operational resilience, but the RMM Users Group gave us a greater appreciation of the issues surrounding resilience. The diversity of perspectives from industry, finance, government, and education helped to associate actual problems with model constructs. Hearing about the real world issues that other organizations had, and how they conquered or planned to conquer them, helped us to be better able to support our own operational teams and to establish a strategy for our organization.*

– Lynn Penn, Director Enterprise Integration, Lockheed Martin Corporation

In the following year, the SEI introduced the concept of a process improvement model for managing operational resilience, drawing heavily upon the SEI's CMMI experience. The SEI continued to collaborate with the FSTC and others to develop an initial framework and subsequent revisions, which resulted in the CERT Resilience Engineering Framework in March of 2008 and v1.0 of the CERT Resilience Management Model (CERT-RMM) in March 2010 (followed shortly thereafter by v1.1 of the CERT-RMM in book form [Caralli 2010a] and a model description in a webinar [Caralli 2010b]). The SEI also developed resilience training and helped establish a CERT-RMM Users Group.[17] The SEI is conducting research and developing resources for measuring operational resilience, including guidance and templates that support organizations in defining their measures and an addendum to CERT-RMM V. 1.1 that updates examples of measures for the 26 process areas [Allen 2011].

16 The FSTC has since been incorporated into the Financial Services Roundtable (http://www.fsround.org).

17 Information on SEI resilience work is available at http://www.cert.org/resilience, including links to the training and the user group pages.

### 4.5.3 The Consequence: Organizations Can Determine Their Capability to Manage Resilience

Organizations in the DoD, the U.S. defense industrial base, U.S. federal civilian agencies, the financial services sector, and academia have been using aspects of the CERT-RMM since 2009. It has been applied to a wide range of problems; some applications are described in podcasts (http://www.cert.org/podcasts). The range of applications includes

- assessing the capability of U.S. IT-based critical infrastructures to be resilient in the presence of attack and the capability of external partners that provide parts of the DoD missions
- building an incident management capability in developing nations
- developing mission assurance planning guides for DoD commanders
- evaluating IT operations and security activities to identify potential improvements and to capture a pre-improvement baseline
- determining whether business continuity policy, when enacted, will produce the intended result
- determining if compliance with mandated regulations results in improved security
- assessing current software development processes to determine if they include software resilience practices
- protecting personally identifiable information and eliminating its use where possible
- measuring operational resilience at strategic and tactical levels

### 4.5.4 The SEI Contribution

The SEI role has been to help organizations institutionalize improved processes for managing operational resilience and measure their benefit, demonstrating the value of converging operational risk disciplines, and accelerating the transition of industrial experience to the broader community. The CERT reputation and leadership role in the information security community and the SEI reputation and leadership role in the process improvement community provide the foundation for this work. The SEI has developed and is transitioning a credible, effective maturity model that allows organizations to have justifiable confidence that they can provide essential services in the presence of disruption and stress and can return to normal operations in a reasonable period of time following disruption.

### 4.5.5 References

[Alberts 1999] Alberts, Christopher; Behrens, Sandra; Pethia, Richard; & Wilson, William. *Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE) Framework, Version 1.0* (CMU/SEI-1999-TR-017). Software Engineering Institute, Carnegie Mellon University, 1999. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=13473

[Allen 2011] Allen, Julia & Curtis, Pamela. *Measures for Managing Operational Resilience* (CMU/SEI-2011-TR-019). Software Engineering Institute, Carnegie Mellon University, 2011. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=10017

[Caralli 2004] Caralli, Richard. *Managing for Enterprise Security* (CMU/SEI-2004-TN-046). Software Engineering Institute, Carnegie Mellon University, 2004. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=7019

[Caralli 2010a] Caralli, Richard A.; Allen, Julia H.; & White, David W. *CERT Resilience Management Model (CERT-RMM): A Maturity Model for Managing Operational Resilience.* Addison-Wesley Professional, 2010 (ISBN 0321712439).

[Caralli 2010b] Caralli, Richard A. "Transforming Your Resilience Management Capabilities: CERT's Resilience Management Model" (webinar). Software Engineering Institute, Carnegie Mellon University, 2010. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=21924

[Joch 2013] Joch, A. "Operational Resilience: Bringing Order to a World of Uncertainty." Federal *Computer Week,* July 8, 2013. http://fcw.com/articles/2013/07/08/exectech-operational-resilience.aspx

## 4.6 Capability Maturity Model Integration

### 4.6.1 The Challenge: Developing a Single Framework for Process Improvement

The success of the various capability maturity models and supporting appraisal methods led the community of users, both in DoD and in industry, to seek a more integrated approach to process improvement. The DoD asked the SEI to develop a model (and associated appraisal method) that would merge the best practices for software development, systems engineering, and acquisition into a single framework that organizations could use for enterprise-wide process improvement initiatives.

**The View from Others**

*Our CMMI Level 5 rating puts us ahead of many of our competitors. This rating demonstrates to our customers that we use proven processes when performing on contracts—and that we are committed to a rigorous process improvement plan to continue to up the ante.*

– Tina Schechter, vice president, Mission Success & Information Technology, for Lockheed Martin MS2 and executive champion for the business unit's CMMI initiative
[Lockheed-Martin 2009]

### 4.6.2 A Solution: The Capability Maturity Model Integration

To improve on the Software Capability Maturity Model (SW-CMM) released in 1993 and subsequent models focused on systems engineering and integrated product and process development, a team of best practice and process improvement experts from government, industry, and the SEI developed the Capability Maturity Model Integration (CMMI) product suite. The update included new process areas, updates to best practices, and generic goals and practices that bring more attention to the planning, definition, measurement, and management of systems and software engineering processes. A *continuous* representation was also created, in addition to the traditional *staged* representation, for organizations that wanted to focus on improvement in certain process areas instead of pursuing an overall maturity rating.

In a capability maturity model, a continuous representation provides many benefits, including a greater degree of granularity in planning and tracking organizational process improvement, and a more revealing look at the trouble spots—and strengths—in organizational practices [CMMI 2010a]. In the continuous representation of CMMI, process areas are organized into categories, such as Process Management, Project Management, Engineering, and Support. Based on its business objectives, an organization selects the process areas in which it wants to improve (e.g., requirements development, risk management, and supplier agreement management) and to what degree. Instead of focusing on maturity levels, the organization uses capability levels (from 0 to 5) to measure improvement relative to each process area. Achievement of a capability level is based on achieving the appropriate specific and generic practices of the selected process area.

In 2002, the CMMI Product Team published the first comprehensive CMMI framework, including models, training, and an appraisal method, which incorporated software, systems engineering, integrated product and process development, and supplier sourcing [CMMI 2002]. The model was rapidly adopted by industry.

#### 4.6.2.1 CMMI Constellations

For CMMI Versions 1.2 and 1.3, improvements were made to the CMMI framework architecture to accommodate the need for multiple CMMI models, while maximizing the use of goals and practices across the different CMMI models. This gave rise to the idea of constellations, in which CMMI models would be derived from careful selections from a larger repository of process areas and practices. As a result, during 2006-2009, CMMI models were developed for product and service development [CMMI 2010a], service establishment, management, and delivery [CMMI 2010b], and product and service acquisition [CMMI 2010c]. Each of these CMMI models was designed to be used in concert with the others, making it easier for organizations to pursue enterprise-wide process improvement. The Standard CMMI Appraisal Method for Process Improvement (SCAMPI) Method Definition Document [SCAMPI 2011] and training materials supported use of the models.

*CMMI for Development (CMMI-DEV).* CMMI for Development was the focus of the initial CMMI framework and was first released in 2002. It was updated in 2006 and again in 2010 with the release of the Version 1.3 model (CMMI-DEV, V1.3). CMMI-DEV describes best practices for the development and maintenance of products and services across their lifecycle. CMMI-DEV combines essential bodies of knowledge, such as software and systems engineering, and dovetails with other process improvement methods that might be used elsewhere in an organization, such as the SEI's Team Software Process (TSP), ISO 9000 [SEI 2009], Six Sigma, and Agile. CMMI-DEV can be used to guide process improvement across a project, division, or organization to lower costs, improve quality, and deliver products and services on time. It is employed by organizations from many industries, including aerospace, banking, computer hardware, software, defense, automobile manufacturing, and telecommunications. For Version 1.3, high maturity process areas were significantly improved to reflect industry best practices, guidance was added for organizations that use Agile methods, and engineering practices and terminology were updated to reflect best practices related to specifying, documenting, and evaluating software architecture.

*CMMI for Acquisition (CMMI-ACQ).* CMMI for Acquisition was first realized as an independent model—the Software Acquisition CMM (SA-CMM) model—in 2002. In 2005, when the SEI and the CMMI Steering Group were planning development of CMMI for Development, Version 1.2, General Motors (GM) approached the SEI about developing a CMMI model that would address acquisition best practices. The SEI had already been directed by the DoD to upgrade the SA-CMM model to be compatible with CMMI. As a recognized leader in IT, GM's vision was to improve how the automaker acquired critical software needed to manage GM's infrastructure around the world. Problems similar to GM's were also being experienced in government acquisition offices. In 2006, the SEI published *Adapting CMMI for Acquisition Organizations: A Preliminary Report* [Hofmann 2006]. This document and its recommendations were piloted and reviewed by acquisition organizations and used as the basis for what became CMMI for Acquisition, Version 1.2, which was later updated to Version 1.3 in 2007. CMMI-ACQ describes practices for acquisition organizations to avoid, eliminate, or mitigate barriers and problems in the acquisition process through improving operational efficiency; initiating and managing the process for acquiring products and services, including solicitations, supplier sourcing, supplier agreement development and award, and supplier capability management; and utilizing a common language for both acquirers

and suppliers so that quality solutions are delivered more quickly at a lower cost with the most appropriate technology. Often, teams must coordinate the functions, manage the risks, and handle information flow as part of a complex relationship with other organizations. CMMI-ACQ provides guidance for this type of challenge.

*CMMI for Services (CMMI-SVC).* The CMMI-SVC model was the first real extension to the CMMI Framework. A model addressing service establishment and delivery was created because the demand for process improvement in services continues to grow (services constitute more than 80 percent of the U.S. and global economy) and stakeholders approached the SEI requesting a model for services. The model covers the activities required to establish, deliver, and manage services. It incorporates work by several service organizations and draws on concepts and practices from other service-focused standards and models, including Information Technology Infrastructure Library (ITIL); ISO/IEC 20000: Information Technology—Service Management; Control Objectives for Information and related Technology (CobiT); and Information Technology Services Capability Maturity Model (ITSCMM).

### 4.6.3 The Consequence: CMMI Models Are Used Effectively Worldwide

CMMI models are being used by small and large organizations alike in a variety of industries, including electronics, health services, finance, government, insurance, and transportation [CMMI Institute 2013]. Adopting organizations include Boeing, General Motors, JP Morgan, Bosch, and many others in North America, Europe, Asia, Australia, and South America. Adoption statistics show the worldwide impact:

- More than 400 organizations are authorized to deliver training and appraisals.
- More than 150,000 professionals have completed the Introduction to CMMI course.
- CMMI appraisals have been reported from 74 countries, and an estimated 2.4 million people work in organizations that have had at least one appraisal since April 2002.

Integration has provided organizations with a number of advantages, including linkage of management and engineering activities to business objectives; the visibility of the product life cycle and engineering activities to ensure that the product or service meets customer expectations; leveraging from additional areas of best practice (measurement, risk management, and supplier management); robust high-maturity practices; the visibility of additional organizational functions critical to their products and services; and coupling with relevant ISO standards [CMMI 2004]. In addition, organizations can more easily pursue enterprise-wide process improvement because each CMMI model in the product suite is designed to be used in concert with other CMMI models.

### 4.6.4 The SEI Contribution

The success of the CMMI project resulted from the contribution of a number of teams from industry, government, and the SEI that worked together to evolve the legacy CMM frameworks into the CMMI framework. The SEI provided the leadership and architectural vision and acted as the steward organization, providing a source of sustainment and continuing support for the adoption, improvement, and evolution of the CMMI product suite. The SEI also worked to ensure the qual-

ity and widespread use of the CMMI and to support its adoption throughout government and industry. In addition, the SEI administered the SEI Partner Network, which in turn provided support for the authorization and maintenance of SCAMPI lead appraisers and instructors for the CMMI introductory course.

In 2012, as part of its mission to transition mature technology to the software community, the SEI transferred CMMI-related products and activities to the CMMI Institute (http://cmmiinstitute.com), a subsidiary of Carnegie Innovations, Carnegie Mellon University's technology commercialization enterprise. The CMMI Institute is working to build upon CMMI's success, advance the state of the practice, accelerate the development and adoption of best practices, and provide solutions to the emerging needs of businesses around the world.

### 4.6.5 References


[CMMI 2002] CMMI Product Team. *CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing, Version 1.1, Continuous Representation (CMMI-SE/SW/IPPD/SS, V1.1, Continuous)* (CMU/SEI-2002-TR-011). Software Engineering Institute, Carnegie Mellon University, 2002. http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=6105

[CMMI 2004] CMMI Product Team. *Upgrading from SW-CMM to CMMI.* Software Engineering Institute, Carnegie Mellon University, Feb 25, 2004. http://resources.sei.cmu.edu/asset_files/WhitePaper/2004_019_001_29417.pdf

[CMMI 2010a] CMMI Product Team. *CMMI for Development, Version 1.3* (CMU/SEI-2010-TR-033). Software Engineering Institute, Carnegie Mellon University, 2010. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9661

[CMMI 2010b] CMMI Product Team. *CMMI for Services, Version 1.3* (CMU/SEI-2010-TR-034). Software Engineering Institute, Carnegie Mellon University, 2010. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9665

[CMMI 2010c] CMMI Product Team. *CMMI for Acquisition, Version 1.3* (CMU/SEI-2010-TR-032). Software Engineering Institute, Carnegie Mellon University, 2010. http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=9657

[SCAMPI 2011] SCAMPI Upgrade Team. *Standard CMMI Appraisal Method for Process Improvement (SCAMPI) A, Version 1.3: Method Definition Document* (CMU/SEI-2011-HB-011). Software Engineering Institute, Carnegie Mellon University, 2011. http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=9703

[CMMI Institute 2013] CMMI Institute. *Results.* http://cmmiinstitute.com/results/ (2013).

[Hofmann 2006] Hofmann, Hubert; Ramani, Gowri; Yedlin, Deborah; & Dodson, Kathryn. *Adapting CMMI for Acquisition Organizations: A Preliminary Report* (CMU/SEI-2006-SR-005). Software Engineering Institute, Carnegie Mellon University, 2006. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=7811

[Kitson 2009] Kitson, David; Vickroy, Robert; Walz, John; & Wynn, Dave. *An Initial Comparative Analysis of the CMMI Version 1.2 Development Constellation and the ISO 9000 Family* (CMU/SEI-2009-SR-005). Software Engineering Institute, Carnegie Mellon University, 2009. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=8849

[Lockheed Martin 2009] "Lockheed Martin's Maritime Systems and Sensors Unit Achieves CMMI Maturity Level 5." Washington, D.C December 1, 2009. http://www.lockheedmartin.com/us/news/press-releases/2009/december/LockheedMartinsMaritimeSy.html

[SEI 2009] *Brief History of CMMI.* Software Engineering Institute, Carnegie Mellon University, 2009. http://resources.sei.cmu.edu/asset_files/Brochure/2009_015_001_28416.pdf

## 4.7 Expanding the CMMI Product Suite to the Acquisition Area of Interest

### 4.7.1 The Challenge: Meeting Acquisition Needs with the CMMI Product Suite

The Capability Maturity Model Integration (CMMI) product suite was spawned from discussions between an SEI sponsor in the Office of the Secretary of Defense (OSD) and the SEI. CMMI was a way to portray integrated approaches to development for both systems and software engineers. Its creation stimulated consideration of companion approaches for other areas of interest that would share many of the core concerns of project management, process management, and support. In the mid-1990s, the U.S. Army requested a model to cover software acquisition [Ferguson 1994].

### 4.7.2 A Solution: CMMI-ACQ – A Full Acquisition Solution

The initial thinking by the OSD sponsor for a CMMI version for acquisition was that a brief approach would be best, and this led to creation of the "Acquisition Module" [Bernard 2005]. The desire for a full CMMI version that allowed both training and assessment of progress came from industry. The CIO of General Motors had orchestrated a new approach to IT software at GM, where all the software that ran each sector's operations would be procured rather than internally developed. But the CIO recognized the value that process discipline would have within his organizations around the world, even though they were acquiring rather than actually developing the software-intensive IT systems under his responsibility. With the SEI's support, a GM team created a draft version that was called a CMMI for Outsourcing [Hoffman 2006]. In 2006, GM completed the draft; and a government, industry, and SEI team was formed to develop the CMMI-ACQ product suite. An advisory board was created to recognize the needs of both government and industry for the final product. Team leadership was shared between an OSD staff member and an SEI project manager. DoD acquisition expertise included two professors at the Defense Acquisi-

**The View from Others**

*At the GAO, we have been using the CMMI-ACQ model to evaluate federal agencies' acquisition efforts. This use of CMMI-ACQ enables the GAO to evaluate acquisition activities across the government using a common methodology.*

– Madhav Panwar, Senior Level Technologist, GAO [Gallagher 2011]

*CMMI for Acquisition (CMMI-ACQ) enables a predictable, consistent, and reliable process for defining the requirements, defining an acquisition strategy, and capturing the best sources.*

– Anthony W. Spehar, VP Missile Systems, Northrop Grumman Aerospace Systems [Gallagher 2011]

*CMMI-ACQ doesn't support the practice of saying 'I'm going to hand this to you, and I'm gone.' Instead, it's about how you interact with your supplier every day to make sure it's done correctly*

– Ralph Szygenda, General Motors CIO, quoted in *Information Week* [Weir 2007]

tion University. Federal agencies, such as the Department of Homeland Security, also participated. A GAO[18] analyst provided insights to ensure that the final product would assist the GAO in its reviews of acquisition programs across the federal domain.

Throughout the development project, the team sought to maintain as much commonality as possible with the CMMI for Development [CMMI Team 2000]. Approximately three-quarters of the model content was virtually identical to the predecessor model. This offers at least two advantages. One is that the commonality often means that understanding is easily transferred from the development domain to its sponsoring agents, the organizations seeking to acquire a well-developed system. The second is that the potential of shared commitment to process improvement by both sides of the contractual relationship offers many potential benefits for teamwork. As some observers had long noted, "a low-maturity acquirer who has contracted with a high-maturity supplier can still deliver lower quality systems to its customers." (This is often caused by ineffective requirements engineering and the resulting "requirements creep.")

The team also recognized that the fit was not exact. A significant portion of an acquisition team member's time is spent creating requests for proposals, reviewing the competitive proposals, selecting a development organization, and then monitoring both the business and technical aspects of the developer's progress, often for several years after contract award. Upon acceptance of the initial products, the acquirer often has to ensure effective transition of the new systems into the business environment. Because many other systems are likely to be affected by the new product's arrival, significant attention to interfaces with them is a particular concern for the acquirer; often only some of the interfaces could be identified within the contractual requirements. Concerns like these, and the variety of contractual mechanisms available, pointed to the need for creating and maintaining an effective acquisition strategy, another difference between the two domains.

**The View from Others**

I believe CMMI-ACQ could have made a considerable difference in the [failed program] and allowed it to continue successfully. Just by reading the purpose of each process area and reflecting on what could have been if the [program] had followed it…, it would undoubtedly be in the Army and Navy inventories today.

— Hon. Claude Bolton, previously Assistant Secretary of the Army [Gallagher 2011]

*When the US Air Force (AF) consolidated various systems engineering assessment models into a single model for use across the entire AF, the effort was made significantly easier by the fact that every model used to build the expanded AF model was based on Capability Maturity Model Integration (CMMI) content and concepts.*

— George Richard Freeman, Technical Director, USAF Center for Systems Engineering [Gallagher 2011]

One of the first pilots of the acquisition model occurred in the international realm. In Australia, the equivalent to the U.S. DoD is called the Defence Materiel Organization, or DMO. One of the

18 At the time this was the General Accounting Office; it is now the Government Accountability Office.

key leaders within that organization wanted to ensure that both Australian suppliers and DMO acquirers would show a commitment to using the best practices captured in CMMI models. They piloted a near-final version and gave the development team some final recommendations that resulted in the release version's being based on some real-world experiences—from half a world away.

### 4.7.3 The Consequence: Acquisition Joins Development for Process Improvement

The acquisition community is able to approach an acquisition with the same discipline that is expected of developers, and has the basis for consistently improving its processes. This not only enables a more predictable acquisition, it sets a reasonable expectation that both parties are committed to following best practices and will identify factors that will improve the target system.

### 4.7.4 The SEI Contribution

The architecture of the CMMI models was based upon foundational work by two SEI Fellows. The notion of a process improvement journey with plateaus of measured accomplishment (staged improvement) was conceived early in the SEI CMM activities [Humphrey 1989]. A companion theory noted that improvement specific to each area of interest might be considered without the breadth of coverage that the staged approach encouraged. This theory resulted in a companion approach for systems engineering [Bate 1994]. Both of these concepts were honored in CMMI, which allows organizational choices or even hybrids to be created if they better stimulate process improvement. The acknowledged leadership of the SEI on these two models and approaches facilitated the development of the teams that brought together government and industry partners and accelerated the transition of the technology to practice.

The SEI created presentation and training materials and worked with standards bodies, including IEEE, and the International Council on Systems Engineering (INCOSE) to further the maturation of the models. Probably most important was the partnership with the National Defense Industrial Association (NDIA) Systems Engineering Committee in finding suitable development team members, along with the SEI sponsors in the DoD. The SEI provided evidence of the viability of the acquisition "variant" through case studies with DoD acquisition organizations, such as the Air Force's Space and Missile Systems Center in Los Angeles. In addition, the SEI Partner Network helped discover potential users in both government and industry around the world. It is significant to note the first full appraisal against the CMMI-ACQ model was performed in a government program office—in Taiwan.

The CMMI-ACQ work is an excellent example of expanding university research by adaption of a successful initial model to satisfy new needs with strong synergies.

### 4.7.5 References

[Bate 1994] Bate, Roger; Reichner, Albert; Garcia-Miller, Suzanne; Armitage, James; Cusick, Kerinia; Jones, Robert; Kuhn, Dorothy; Minnich, Ilene; Pierson, Hal; & Powell, Tim. *A Systems*

*Engineering Capability Maturity Model, Version 1.0* (CMU/SEI-94-HB-004). Software Engineering Institute, Carnegie Mellon University, 1994. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=12037

[Bernard 2005] Bernard, Thomas; Gallagher, Brian; Bate, Roger; & Wilson, Hal. *CMMI Acquisition Module (CMMI-AM), Version 1.1* (CMU/SEI-2005-TR-011). Software Engineering Institute, Carnegie Mellon University, 2005. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=7665

[CMMI Team 2000] CMMI Product Development Team. *CMMI for Systems Engineering/Software Engineering, Version 1.02, Staged Representation (CMMI-SE/SW, V1.02, Staged)* (CMU/SEI-2000-TR-018). Software Engineering Institute, Carnegie Mellon University, 2000. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=5263

[Ferguson 1994] Ferguson, Jack; Cooper, Jack; Falat, Michael; Fisher, Matthew; Guido, Anthony; & Marciniak, John. *Software Acquisition Capability Maturity Model* (CMU/SEI-96-TR-020). Software Engineering Institute, Carnegie Mellon University, 1996. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=12619

[Gallagher 2011] Gallagher, Brian; Phillips, Mike; Richter, Karen; & Shrum, Sandy. *CMMI for Acquisition*. Addison-Wesley Professional, 2011 (ISBN 0321711513).

[Hoffman 2006] Hoffman, Hubert; Yedlin, Deborah; Mishler, John; & Kushner, Susan. *CMMI for Outsourcing: Guidelines for Software, Systems, and IT Acquisition*. Addison-Wesley Professional, 2007 (ISBN 0321477170).

[Humphrey 1989] Humphrey, Watts S. *Managing the Software Process.* Addison-Wesley Professional, 1989 (ISBN 0201180952). http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=30884

[Weir 2007] Weier, Mary Hayes. “General Motors CIO Promotes Procurement Standards.” *Information Week.* November 8, 2007. http://www.informationweek.com/general-motors-cio-promotes-procurement-standards/d/d-id/1061282?

## 4.8 The Smart Grid

### 4.8.1 The Challenge: The Need for New Approaches for Utilities

Demand for electricity worldwide is projected to nearly double from 17,200 terawatt hours (TWh)[19] in 2009 to over 31,700 TWh in 2035 [PwC 2012]. Prices for electricity in Western countries are predicted to increase by 400 percent in 30 years [NRG 2012]. Increased demand could also lead to more blackouts. A PricewaterhouseCoopers study projects that blackouts in North America and Europe are two to three times more likely to occur by 2030 [PwC 2012]. Sustainable sources would reduce carbon emissions from electricity generation, yet non-fossil fuels provide the source for only 34 percent of electricity generation today.

A smart grid helps to address some of these issues, and transformation to smart grid is a major undertaking. Thus, electric utilities must carefully consider the reasons to invest in it. Some utilities might be driven to a smart grid transformation to protect against a steep rise in electricity generation and delivery costs as energy consumption explodes in the coming decades. Other utilities might want to push forward with sustainable sources to reduce carbon emissions from electricity generation [NRG 2012]. Still other utilities might desire to build empowered and involved workforces, improve business performance, create greater customer satisfaction, extend asset life, or comply with regulations [SGMM Team 2010a]. Whatever the motivation, industry consultants advise utilities to recognize the need to "define a smart grid vision and develop a road map to get there" [Asthana 2010]. As Steve Rupp of SAIC Energy, Environment and Infrastructure says, "The key to success in any grid transformation is to have a good plan and to work that plan" [Rupp 2012].

### 4.8.2 A Solution: Smart Grid Model and Transformation Process

Smart Grid Maturity Model (SGMM) development began in 2007, when IBM formed a coalition of major utility companies, the Global Intelligent Utility Network Coalition (GIUNC).[20] IBM, GIUNC, and American Productivity and Quality Center (APQC) created the model to change the way power is generated, distributed, and used by adding digital intelligence to the current systems. The SGMM supports the transformation process and helps utilities with planning. In 2009, IBM handed off SGMM stewardship to the SEI because it believed that a neutral third party would be more effective in encouraging industry adoption of the model [Jones 2009]. With input from industry stakeholders and the Department of Energy (DOE)—the stewardship sponsor—the SEI released Version 1.1 of the SGMM in 2010 [SGMM Team 2010b] and Version 1.2 in 2011 [SGMM Team 2011].

The SGMM product suite consists of the model itself, the Navigation Process of expert-led workshops and analysis, the Compass questionnaire-based assessment for determining maturity ratings

19 Wh = terawatt hours

20 The GUINC website is https://www-304.ibm.com/communities/service/html/communityview?communityUuid=1a988236-4f84-4a80-8d8b-b5a288d1566a. A 2009 press release can be seen at www-03.ibm.com/press/us/en/pressrelease/28838.wss.

and performance comparisons, Navigation Process training, and a program to license organizations and certify individuals to deliver the Navigation Process. The *SGMM Matrix* offers a summary view of the model domains and expected characteristics in each domain and at each maturity level.[21]

**The View from Others**

*Pepco Holdings has been involved with the SGMM since its inception. We recently completed the survey again, using the SGMM Navigation process. This was helpful in fostering candid, fact-based discussion of where we have been, where we are today, and where we expect to be in the future. We look forward to using the tool as an integral part of our ongoing planning and transformation process, and in measuring our progress over time.*

– George Potts, Vice President, Business Transformation, Pepco Holdings, Inc. [SGMM 2010a]

### 4.8.3 The Consequence: Effective Method for Utilities' Transition to the Smart Grid

Utilities that have approached their smart grid transformation planning with the SGMM Navigation Process have seen a number of benefits. They have the evidence needed to maintain financial support and an initial opportunity to formally review and plan their smart grid activities. The use of a common language across the enterprise enables effective communication about smart grid realities and objectives.

Ultimately, in the U.S. and other nations, grid reliability, security, efficiency, and safety will increase. As organizations progress, they manage power flows so that power losses are minimized and the usage of lowest cost power generation resources are maximized. They implement business processes that deliver an environmentally friendly energy network while minimizing costs and sustaining profitability. The growing number of industry "SGMM Navigators," trained by the SEI, means there is model expertise in industry able to guide the SGMM Navigation Process.

### 4.8.4 The SEI Contribution

Initial work leading to the SGMM was done by an IBM-led coalition of major utility companies, the Global Intelligent Utility Network Coalition. Their work was handed over to the SEI as a neutral, third-party steward. The Department of Energy funded the SEI to take on the SGMM stewardship. In addition to its neutrality, the SEI had an existing relationship and other work in critical infrastructure. The DOE provides tools in support of public-private partnership efforts to modernize the grid as a national priority. The SEI is also collaborating with APQC, a non-profit member-based research organization to help organizations adopt the model.

As SGMM steward, the SEI evolves the model and makes it freely available as a service to the utility industry. With industry stakeholder and DOE input, the SEI released Version 1.1 of the model in 2010 and Version 1.2 in 2011. In addition, the SEI developed the SGMM product suite that includes the Navigation Process, the Compass assessment, Navigation Process training, and the licensing and certification program for delivering the Navigation Process. SGMM expertise is

21 SGMM products and other SGMM information are available at http://www.sei.cmu.edu under "Work Areas."

becoming more widespread. As utilities adopt the smart grid, U.S. and other nations' grid reliability, security, efficiency, and safety will increase.

### 4.8.5 References


[Asthana 2010] Asthana, Anjan; Booth Adrian; & Green, Jason. *Best Practices in the Deployment of Smart Grid Technologies.* McKinsey and Company, 2010.

[Jones 2009] Jones, K. C. "Carnegie Mellon to Oversee IBM Smart Grid Maturity Model." *InformationWeek* March 30, 2009. http://www.informationweek.com/carnegie-mellon-to-oversee-ibm-smart-gri/216401730

[NRG 2012] NRG Experts. "Where We've Been, Where We're Going: NRG's Smart Grid FAQs and Primer." March 30, 2012. http://www.smartgridnews.com/artman/publish/Business_Markets_Pricing/NRG-s-smart-grid-FAQs-and-primer-4623.html

[PwC 2012] PricewaterhouseCoopers (PwC). *The Shape of Things to Come: Investment, Affordability, and Security in an Energy-Hungry World.* 12th PwC Annual Global Power & Utilities Survey, 2012.

[Rupp 2012] Rupp, Steven S. *The California Energy Commission and SGMM: Partners for a Future Vision of Smart Grid* (webinar). http://www.sei.cmu.edu/library/abstracts/presentations/webinar20120321.cfm (2012).

[SGMM Team 2010a] SGMM Team. *Smart Grid Maturity Model: Update October 2010.* Software Engineering Institute, Carnegie Mellon University, 2010. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=28296

[SGMM Team 2010b] SGMM Team. *Smart Grid Maturity Model, Version 1.1: Model Definition* (CMU/SEI-2010-TR-009). Software Engineering Institute, Carnegie Mellon University, 2010. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9455

[SGMM Team 2011] SGMM Team. *Smart Grid Maturity Model, Version 1.2: Model Definition* (CMU/SEI-2011-TR-025). Software Engineering Institute, Carnegie Mellon University, 2011. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=10035

## 4.9 Software Risk Management

### 4.9.1 The Challenge: Assessing and Managing Software Risks

Although the DoD (and NASA) had mature systems risk management approaches in the early 1990s, software risks were largely ignored, partially because there was no effort to collect and categorize sources of previous failures. The DoD continued to experience highly publicized failures and faced an acknowledged inability to take full advantage of the potential benefits that software offered [Dick 1991].

As the SEI's technical reputation grew, the DoD began requesting that the SEI conduct "red-team" assessments on programs that appeared to be in serious trouble. After one such exercise, the DoD program manager asked the SEI to provide a briefing of its findings. After the presentation, a senior DoD official in the audience issued a challenge: "It is clear that an SEI team of software experts can successfully assess the risks in a software system. Would it be possible for the SEI to capture that knowledge and present it in a way that knowledgeable people who are not experts can make a similar assessment without the SEI's help?" The SEI realized that if it could rise to this challenge, the community's understanding of the causes of failure would grow over time.

### 4.9.2 A Solution: Apply Risk Management Techniques to Software

The SEI response was to apply the discipline and techniques of risk management to the acquisition and development of large software-intensive systems. The SEI proposed to the DARPA program manager a new effort in software risk management. At the time, the Air Force was having difficulty with upgrades to the C-17, and the House Armed Services Committee suggested that the SEI should help. DARPA agreed to SEI's proposal with the understanding that the SEI would help the C-17 program identify and mitigate risks in its process. With the support of the House Armed Services Committee, additional funds were provided to pursue this approach.

The initial SEI effort was to establish a community effort through a series of workshops leading up to a Risk Conference, held jointly with the National Security Industrial Association (NSIA) in October 1991. Over the next several years, SEI risk research was focused on risk identification. One of the earliest publications was the Software Risk Taxonomy [Carr 1993], which documented sources of risk that decision makers should consider when identifying risks. It also documented the early version of the first risk management method—the Software Risk Evaluation (SRE)—a structured method to identify and analyze the risks on a software program.

SEI researchers continued to refine the SRE method as they worked with the Navy Program Executive Officer (PEO)(A) and the Navy Looking Glass program. A key achievement at this time was the development of training in risk identification and analysis for SRE team members. The SEI developed training to ensure that interviewers would ask questions in a consistent, non-threatening, non-judgmental, and non-leading manner. An additional aspect was investigating how acquisition programs might be able to manage risks in collaboration with contractors to enable a program manager to gain a clear picture of all the program's risks. The research and lessons learned from working with the Navy led to many of the concepts embodied in what became Continuous Risk Management (CRM) and Team Risk Management (TRM). Additional work with NASA led to the production of the *Continuous Risk Management Guidebook* in 1996 [Dorofee 1996] and the

Continuous Risk Management training course. The guidebook and associated course enabled program managers and risk managers to learn how to manage risks more effectively.

In 1997, the SEI began to broaden its software risk management approach to other software-related areas including cybersecurity. When the Army wanted a risk management approach tailored to acquisition programs, this work became the foundation for guidance in implementing the risk management process area of the Software Acquisition CMM. The COTS Usage Risk Evaluation (CURE) and the Architecture Tradeoff Analysis Method (ATAM), which focused on COTS products and software system architecture, respectively, were variations on the original risk assessment (SRE). Risk was incorporated into CMMI in 2000 as a practice area (RSKM) in CMMI V1.02.

Risk management was also a focus in the ongoing research into cybersecurity by the CERT Coordination Center (CERT/CC). In 1997, the Information Security Evaluation (ISE), a variation of the SRE, was used to identify vulnerabilities in operational, networked information technology systems. In 1998, CERT researchers began developing a new approach for managing cybersecurity risks within an organization based on the principles of CRM and ISE. This research and the Defense Health Information Assurance Program (DHIAP)[22] were the driving forces for developing the Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE) [Alberts 2003]. The goal was to develop a self-directed risk assessment as part of the DoD effort to comply with the data security requirements defined by the Health Insurance Portability and Accountability Act (HIPAA) of 1996. DHIAP transitioned OCTAVE to the Air Force and Army in 2001, using the SEI OCTAVE training. OCTAVE continues to be a widely used information security risk assessment method.

**The View from Others**

*I think the biggest contribution was to bring awareness to the subject [of software risk management], help legitimize it as a program/project management concern, and gave a process for operationalizing it in a useful way.*

– Robert Charette, founder of ITABHI Corporation and Chairman of the SEI Risk Program Advisory Board

A Navy Program Manager stated in the late 1990s that all of his investment in the SEI Risk Program had paid off with the identification and mitigation of a single catastrophic risk.

By 2005, software acquisition and development programs were becoming more distributed in nature, often comprising multiple geographically distributed organizations. Traditional risk approaches did not readily scale to these networked, highly complex program environments. In 2006, the SEI began research into managing risks in interactively complex software-reliant systems. This led to new methods for assessing risk and success factors in complex networked systems (e.g., Mission (Risk) Diagnostic [Alberts 2008, 2009, 2012]) and a focus on using key drivers of success to produce a systemic view of program risk

22 DHIAP was a small consortium of organizations, including the SEI and the Advanced Technology Institute (ATI) of the South Carolina Research Authority (SCRA), overseen by a group from the Telemedicine Advanced Technology Research Center (TATRC) from Fort Detrick, Maryland.

[Alberts 2009]. In 2010, SEI researchers began to apply these new risk principles as part of a research effort into developing a method for assessing risk in the software supply chain. Finally, much of the SEI's current risk management work is focused on software assurance. In 2014, SEI researchers began developing the Security Engineering Risk Analysis (SERA) method, a systematic risk-based method for building security into software-reliant systems rather than deferring security to later lifecycle activities such as operations.

### 4.9.3 The Consequence: A Disciplined Approach to Identifying and Managing Software Risks

The SEI had a significant impact on the community in terms of risk management, primarily by establishing the foundation of a defined practice and systematic way of identifying and codifying risks. The SEI risk research produced one of the standards for software risk management, enabling program managers in all types of software-intensive programs to do a better job of identifying what could go wrong and mitigating the worst of those risks. In the Cutter Consortium's report, *The State of Risk Management 2002*, 21 percent of respondents to a survey about risk management techniques said that they use SEI standards for risk management. Only ISO ranked higher, with 36 percent of respondents.[23]

### 4.9.4 The SEI Contribution

The SEI software risk management effort benefited from broad community input. Working with the Department of Defense, NASA, industry, and cybersecurity experts and managers provided a wealth of useful techniques and lessons learned, as well as the opportunities to improve different approaches to solving the problems associated with risk management. Without these contributions, the resulting methods and approaches of the SEI's work would not be as rich, deep, and broad.

The SEI risk research continues today, examining specific problems associated with today's highly complex, interdependent programs and finding new ways to deal with the emergent issues of tomorrow.

### 4.9.5 References

[Alberts 2003] Alberts, Christopher & Dorofee, Audrey. *Managing Information Security Risks: The OCTAVE Approach*. Addison-Wesley Professional, 2003 (ISBN 03211188630).

[Alberts 2008] Alberts, Christopher; Dorofee, Audrey; & Marino, Lisa. *Mission Diagnostic Protocol, Version 1.0: A Risk-Based Approach for Assessing the Potential for Success* (CMU/SEI-2008-TR-005). Software Engineering Institute, Carnegie Mellon University, 2008. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=8665

23 Detailed information on this report is available only to registered Cutter users. However, some information is available at http://www.cutter.com/cgi-bin/search/usr/local/etc/httpd/htdocs?filter=&query=hype+or+reality.

[Alberts 2009] Alberts, Christopher & Dorofee, Audrey. *A Framework for Categorizing Key Drivers of Risk* (CMU/SEI-2009-TR-007). Software Engineering Institute, Carnegie Mellon University, 2009. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9093

[Alberts 2012] Alberts, Christopher & Dorofee, Audrey. *Mission Risk Diagnostic (MRD) Method Description* (CMU/SEI-2012-TN-005). Software Engineering Institute, Carnegie Mellon University, 2012. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=10075

[Carr 1993] Carr, Marvin; Konda, Suresh; Monarch, Ira; Walker, Clay; & Ulrich, Carol. *Taxonomy-Based Risk Identification* (CMU/SEI-93-TR-006). Software Engineering Institute, Carnegie Mellon University, 1993. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=11847

[Dick 1991] Dick, David R., Col. USAF. “Slaying the Software Dragon” (Memo 91-02423). Armed Forces Communications and Electronics Association, June 1, 1991.

[Dorofee 1996] Dorofee, Audrey; Walker, Julie; Alberts, Christopher; Higuera, Ronald; Murphy, Richard; & Williams, Ray. *Continuous Risk Management Guidebook*. Software Engineering Institute, Carnegie Mellon University, 1996. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=30856

## 4.10 Personal Software Process and Team Software Process

### 4.10.1 The Challenge: Improving Software Quality During Development

As the use of computers and the development of software grew in the 1960s and 1970s, it was accompanied by growing pains: many projects failed to deliver quality products within a predictable time or budget and project failure was common [Humphrey 1989]. In response to a request from the Department of Defense, the SEI led the development of the Capability Maturity Model (CMM), which captured organizational best practices for software development [Paulk 1993]. *Maturity* relates to the degree of formality and optimization of processes, from ad hoc practices, to formally defined steps, to managed result metrics, to active optimization of the processes. When the model was applied to an existing organization's software development processes, it provided an effective approach toward improving them.

Although the CMM began to see widespread adoption, some problems remained. An early misperception of the CMM was that it did not apply to small organizations or projects. Another issue was that the CMM told people *what* to do, it did not help them understand *how*, and software development practice was nearer a craft than an engineering discipline. Most finished software products could be made to work, but only after extensive testing and repair. And as software programs grew larger and larger, the difficulty of finding and fixing problems also began to increase exponentially.

### 4.10.2 A Solution: Personal Software Process and Team Software Process

Because he believed software quality starts with the individual engineer, an SEI Fellow decided to apply the underlying principles of the CMM to the software development practices of a single developer. From 1989 to 1993, he wrote more than 60 programs and more than 25,000 lines of code using CMM practices and concluded that the management principles embodied in the CMM were just as applicable to individual software engineers. The resulting process was the Personal Software Process (PSP) [Humphrey 1994].

The PSP is a structured software development process that helps software engineers understand and improve their performance by using a disciplined, data-driven procedure. It includes effective defect management techniques and comprehensive planning, tracking, and analysis methods. PSP training follows an evolutionary improvement approach: engineers learning to integrate the PSP into their processes beginning at the first level and progressing in process maturity to the final level. Each level has detailed scripts, checklists, and templates to guide engineers through required steps that help individual engineers improve their own personal software process. Properly used, the PSP provides the historical data engineers need to better make and meet commitments.

It soon became obvious that, while excellent results were possible using the PSP, it was almost impossible to maintain the discipline required for PSP practices if the surrounding environment did not encourage and demand them. Systems development is a team activity, and the effectiveness of the team largely determines the quality of the engineering. A process for the smallest operational unit in most organizations, the project team, called Team Software Process (TSP) was designed in 1996 [Humphrey 2000].

The principal motivator for the development of the TSP was the conviction that engineering teams could do extraordinary work, but only if they were properly formed, suitably trained, staffed with skilled members, and effectively led. The objective of the TSP is to build and guide such teams. The TSP software development cycle begins with the launch, a planning process led by a specially trained coach. The launch is designed to begin the team-building process, and during this time teams and managers establish goals, define team roles, assess risks, estimate effort, allocate tasks, and produce a team plan. During an execution phase, developers track planned and actual effort, schedule, and defects, meeting regularly (usually weekly) to report status and revise plans. An important element of the TSP is the measurement framework. Engineers using the TSP collect three basic measures: size, time, and defects. They use many other measures that are derived from these three basic measures. The measurement framework consolidates individual data into a team perspective. The data collected are analyzed weekly by the team to understand project status against schedule and quality goals. A development cycle ends with a post mortem to assess performance, revise planning parameters, and capture lessons learned for process improvement.

### 4.10.3 The Consequence: Improved Quality at the Individual and Team Levels

Experience with the TSP has shown that it improves the quality and productivity of engineering teams while helping them to more precisely meet cost and schedule commitments. A study undertaken in 2003 demonstrated that teams using the TSP were able to meet critical business needs by delivering essentially defect-free software on schedule and with better productivity. While industry data indicated that over half of all software projects were more than 100 percent late or were cancelled, the 20 TSP projects in 13 organizations included in the study delivered their products an average of 6 percent later than they had planned. These TSP teams also improved their productivity by an average of 78 percent. The teams met their schedules while producing products that had 10 to 100 times fewer defects than typical software products. They delivered software products with average quality levels of 5.2 sigma, or 60 defects per million parts (lines of code). In several instances, the products delivered were defect free [Davis 2003].

**The View from Others**

*Our schedule reliability is now +/- 10 percent from –50/+200 percent and our defect density at the team level has been reduced by over 50 percent.*

*One of my first projects as an embedded systems programmer finished on the day we planned to finish six months earlier. I attribute the success to planning at a better granularity and making full use of the earned value tracking. The day we got 100 percent earned value was the day we planned to get 100 percent value, and we as a team celebrated like we had won a basketball game.*

*Multiple projects in our organization have been able to keep within their time schedules (+/- three weeks) over a six-month span. This is something we [had] not been able to accomplish in the past. This is one of the reasons that management is very happy with the TSP process.*

These quotes are from a team that attended the PSP for Engineers course and used PSP in its organization to meet TSP goals [Davis 2003].

An in-progress study of 214 TSP projects provides additional evidence of the benefits of disciplined practice. The average CPI (Cost Performance Index) for these projects was 0.93, the average SPI (Schedule Performance Index) was 0.88, and the average System Test Defect Density was 1.32 defects per KSLOC (1000 lines of code, or LOC).

Specific examples of improvements include these:

- Hill Air Force Base, near Salt Lake City, Utah, is the first U.S. government organization to be rated at CMM Level 5. The first TSP project at Hill found that team productivity improved 123 percent and test time was reduced from an organizational average of 22 percent to 2.7 percent of the project schedule.
- Boeing, on a large avionics project, had a 94 percent reduction in system test time, resulting in a substantial improvement in the project schedule and allowing Boeing to deliver a high-quality product ahead of schedule [Davis 2003].
- Teradyne found that, prior to the TSP, defect levels in integration test, system test, field testing, and customer use averaged about 20 defects per KLOC. The first TSP project reduced these levels to 1 defect per KLOC. Since it cost an average of 12 engineering hours to find and fix each defect, Teradyne saved 229 engineering hours for every 1000 LOC of program developed.
- Advanced Information Services reported in 2012 that its use of the TSP continues to result in systems with very predictable schedule and quality. The company is currently averaging 0.3765 defects per KSLOC during user acceptance testing. In fact, quality and schedule are so predictable with TSP that the company is able to support fixed price contracts that include a warranty against defects after user acceptance test. In 2011, Advanced Information Services delivered a large, 570 KSLOC software application to the Selective Service System with a delivered defect density of 0.097 defects/KSLOC [Sheshagari 2012, Ratnaraj 2012].
- Beckman Coulter reported in 2012 that first-time use of the TSP resulted in 5 to 100 times improvement in fielded software on six different medical devices [Van Eps 2012].

A major contributor to the success of TSP teams, besides data, is the commitment and ownership generated during the launch and sustained throughout the life of the project. It is the synergy that is created when a team has a common goal and each and every person on that team understands how his or her work and everyone else's work contributes to the achievement of that goal.

### 4.10.4 The SEI Contribution

If even the smallest programs are not of the highest quality, they will be hard to test, take time to integrate into larger systems, and be cumbersome to use. The SEI was an early contributor to the idea that software could be significantly improved, from the bottom up, by bringing discipline to the performance of individual engineers and engineering teams. Only a statistically managed software engineering discipline can support the growing size and complexity of today's systems. PSP and TSP provide the framework and data required.

### 4.10.5 References

[Carleton 2010] Carleton, Anita; Over, James; Schwalb, Jeff; Kellogg, Delwyn; & Chick, Timothy. *Extending Team Software Process (TSP) to Systems Engineering: A NAVAIR Experience Report* (CMU/SEI-2010-TR-008). Software Engineering Institute, Carnegie Mellon University, 2010. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=9443

[Davis 2003] Davis, Noopur & Mullaney, Julia. *The Team Software Process (TSP) in Practice: A Summary of Recent Results* (CMU/SEI-2003-TR-014). Software Engineering Institute, Carnegie Mellon University, 2003. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=6675

[Humphrey 1989] Humphrey, Watts S. *Managing the Software Process.* Addison-Wesley Professional, 1989 (ISBN 0201180952). http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=30884

[Humphrey 1994] Humphrey, Watts S. *A Discipline for Software Engineering: The Complete PSP Book.* Addison-Wesley Professional, 1994 (ISBN 0201546108). http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=30873

[Humphrey 1997] Humphrey, Watts S. *Managing Technical People: Innovation, Teamwork, and the Software Process.* Addison-Wesley Professional, 1997 (ISBN 0201545977). http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=30850

[Humphrey 2000] Humphrey, Watts S. *Introduction to the Team Software Process.* Addison-Wesley, 2000 (ISBN 020147719X). http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=30774

[Paulk 1993] Paulk, Mark; Curtis, William; Chrissis, Mary Beth; & Weber, Charles. *Capability Maturity Model for Software (Version 1.1)* (CMU/SEI-93-TR-024). Software Engineering Institute, Carnegie Mellon University, 1993. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=11955

[Ratnaraj 2012] Ratnaraj, David Y. “Excellence: Methodology Assists, Discipline Delivers.” *TSP Symposium.* St. Petersburg, FL, September 17-20, 2012. Software Engineering Institute, Carnegie Mellon University, 2010. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=298076

[Sasao 2010] Sasao, Shigeru; Nichols, William; & McCurley, James. *Using TSP Data to Evaluate Your Project Performance* (CMU/SEI-2010-TR-038). Software Engineering Institute, Carnegie Mellon University, 2010. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9677

[Sheshagari 2012] Sheshagari, Girish. “High Maturity Practices, The Way Forward: Solving Software Engineering’s Persistent Problems.” *TSP Symposium*. St. Petersburg, FL, September17-20, 2012. Software Engineering Institute, Carnegie Mellon University, 2012. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=298012

[Van Eps 2012] Van Eps, Scott & Marshall, Rick. “Using the TSP at the End of the Development Cycle.” *TSP Symposium.* St. Petersburg, FL, September 17-20, 2012. Software Engineering Institute, Carnegie Mellon University, 2012. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=298042

## 4.11 Measurement and Analysis

### 4.11.1 The Challenge: Measuring Software Development Capabilities and Products

Measurement and analysis in software engineering has long been a topic of interest. Without it, there is no clear, quantitative picture of software development capabilities or a basis for predicting or comparing products and processes. Although early software projects used several systems for predicting software costs, measurement was done in different ways and was often based on definitions that were inconsistent. The need for a standard and reliable set of measures that would help acquisition program managers and software development contractors alike was an early priority for the SEI.

### 4.11.2 A Solution: Approaches for Collecting and Analyzing Data

In response to the DoD's 1991 Software Technology Strategy, the SEI agreed to lead the development of a set of core measures to "help the DoD plan, monitor, and manage its internal and contracted software development projects" [Carleton 1992]. In collaboration with measurement experts, including those who developed prediction systems, the SEI developed definition frameworks for a set of core measures. The measures focus on size, defects, effort, and schedule. These definition frameworks make it possible for organizations to use the measures that best match their processes and infrastructure while benefiting from a standard way of describing the operational definitions in detail.

Once the definition problem for measures was resolved, the SEI turned its attention to helping organizations decide what to measure. Experience from management information systems shows that many reports could be generated that have little to no impact on decision making within the organization. The SEI began to investigate the goal-question-metric method for aligning measurement with information needs in the organization [Basili 1984]. The SEI modified this approach to include explicit consideration of the output of the measurement activity—that is, the *indicator* to be used by decision makers. The method was dubbed goal-question-indicator-measure (GQ(I)M). Using the GQ(I)M approach helps mitigate the risk of measuring and reporting information that provides little or no value to the organization.

As approaches for conducting measurement effectively matured and were disseminated, the next significant challenge became data analysis [Paulk 2000]. The notion of analysis, especially analysis related to process improvement, was often equated with the high-maturity practices of the SW CMM and CMMI. More focus on the "analysis" part of measurement and analysis was also spurred by the establishment of the Measurement and Analysis process area in the CMMI [CPT 2002].

An early and foundational work in this area was *Measuring the Software Process*, which showed how statistical process control techniques could be fruitfully applied to software data [Florac 1999]. This was followed by substantial work on the application of Six Sigma analytical techniques to software engineering [Penn 2007]. The Six Sigma connection provided a rich set of tools as well as a "brand" that already had roots in many organizations, facilitating its adoption. The current work on developing estimates early in the DoD acquisition lifecycle incorporates

techniques used in Six Sigma, such as Bayesian Belief Networks, matrix transformation, and subjective input calibration methods. The method quantifies uncertainties, allows subjective inputs, visually depicts influential relationships among program change drivers and outputs, and assists with the explicit description and documentation underlying an estimate.

### 4.11.3 The Consequence: Effective, Quantitative Basis for Improvement

The SEI work in measurement and analysis has had far-reaching impact. The core measures report became a foundational work and is referenced by many subsequent publications on software measurement, and the definition checklists have been widely incorporated as part of the approach for specifying measures for cost estimation. SEI-developed training courses on analyzing software data help others leverage the power of statistics in understanding and gaining insight from the measures collected about software projects, processes, and products.

### 4.11.4 The SEI Contribution

The SEI did not work alone in its attempt to move the measurement and analysis community forward. One specific application of the early measurement definition work was in conjunction with the Cost Constructive Cost Model (CoCoMo) [Boehm 2000]. A primary input to cost models is an estimate of the size of the software to be built. The CoCoMo model uses the SEI's size definition checklist approach to specify the operational definition of the number of lines of code to be developed. Also, much work, especially in the early years, was done in collaboration with the Practical Software Measurement (PSM) initiative, sponsored by the DoD and the U.S. Army [PSM 2012]. While there are some differences in methods and techniques, the underlying principles in SEI and PSM measurement and analysis products are virtually the same. The SEI continues to participate in the PSM's active, collaborative forum for measurement and analysis.

The SEI has consistently conducted research and development related to measurement and analysis. While the initial focus was on helping to identify and standardize measures related to project management and process improvement, it grew to include the application of quantitative analytical techniques for use of the data. Looking to the future, the SEI is developing a research agenda to investigate the application of probabilistic and modeling techniques in measurement and analysis.

### 4.11.5 References

[Basili 1984] Basili, V. & Weiss, D. "A Methodology for Collecting Valid Software Engineering Data." *IEEE Transactions on Software Engineering 10*, 3 (November 1984): 728-738.

[Boehm 2000] Boehm, Barry W.; Abts, Chris; Brown, Winsor A.; Chulani, Sunita, et al. *Software Cost Estimation With Cocomo II*. Prentice Hall, 2000 (ISBN 0130266922).

[Carleton 1992] Carleton, Anita; Park, Robert; Bailey, Elizabeth; Goethert, Wolfhart; Florac, William; & Pfleeger, Shari. S*oftware Measurement for DoD Systems: Recommendations for Initial Core Measures* (CMU/SEI-92-TR-019). Software Engineering Institute, Carnegie Mellon University, 1992. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=11675

[CPT 2002] CMMI Product Team. *CMMI for Software Engineering, Version 1.1, Staged Representation (CMMI-SW, V1.1, Staged)* (CMU/SEI-2002-TR-029). Software Engineering Institute,

Carnegie Mellon University, 2002. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=6217

[Florac 1999] Florac, William A. & Carleton, Anita D. *Measuring the Software Process: Statistical Process Control for Software Process Improvement.* Addison-Wesley Professional, 1999 (ISBN 0201604442).

[Paulk 2000] Paulk, Mark & Chrissis, Mary Beth. *The November 1999 High Maturity Workshop* (CMU/SEI-2000-SR-003). Software Engineering Institute, Carnegie Mellon University, 2000. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=5013

[Penn 2007] Penn, M. L.; Siviy, Jeannine; & Stoddard, Robert W. *CMMI and Six Sigma: Partners in Process Improvement.* Addison-Wesley Professional, 2007 (ISBN 0321516087).

[PSM 2012] Practical Software and Systems Measurement. "Practical Software and Systems Measurement: A Foundation for Objective Project Management." http://www.psmsc.com/AboutPSM.asp (2012).

## 4.12 Developing a Measurement System That Supports an Organization's Goals

### 4.12.1 The Challenge: Software Project Measurements That Support Business Goals

Despite significant improvements in implementing measurement programs for software development, a large percentage of measurement programs are not successful. Organizations often do not achieve the potential benefits of a sound measurement program due to the inconsistent construction and interpretation of indicators derived from measurement data. One of the dangers in enterprises as complex as software development and support is that there are potentially so many things to measure that users are easily overwhelmed by the opportunities. The search for the "right" measures can easily become confusing when the selection is not driven by the information requirements to be addressed by the measures. A successful measurement program is more than collecting data. The benefits and value of doing software measurement comes from the decisions and actions taken in response to analysis of the data, not from the collection of the data. The SEI was challenged to develop a measurement and analysis methodology to support the goals of an organization and to ensure that data is not collected for the sake of collection alone.

### 4.12.2 A Solution: Goal-Driven Software Measurement—Goal-Question-Indicator

To address the challenge, the goal-question-metric (GQM) methodology, introduced and described by Basili and Rombach[24] [Basili 1988, Rombach 1989], was enhanced and augmented by the SEI into the goal-question-(indicator)-metric methodology (GQIM), a disciplined approach to defining a set of measures and indictors related to the goal. The goal-driven software measurement process produces measures that provide insights into important management issues as identified by the business goals. Since the measurements are traceable back to the business goals, the data collection activities stay better focused on their intended objectives. In goal-driven measurement, the primary question is not "What metrics should I use?" but "What do I want to know or learn?" [Rombach 1989].

The steps of the approach are organized into three sets of activities: identifying goals, defining indicators and the data needed to produce them, and creating an action plan to guide the implementation. Business goals are translated into measurement goals [Basili 1984, Briand 1996] by refining them into concrete, operational statements with a measurement focus. This refinement process involves probing and expanding each high-level goal to derive questions. The questions provide concrete examples that can lead to statements that identify what type of information is needed. From these questions, displays or indicators are postulated that provide answers and help link the measurement data that will be collected to the measurement goals. The goal-driven approach requires that indicators (charts, tables, or other types of displays and reports) be sketched out and approved by the intended user. These indicators serve as a requirements specification for the data that must be gathered, the processing and analysis that must take place, and the schedule for these

24 Basili, Victor R. "Using Measurement for Quality Control and Process Improvement." 2nd Annual SEPG Workshop. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA., June 21-22, 1989. No longer available.

activities. The final set of activities uses the output of the preceding two sets of activities to develop an action plan.

The indicator template developed by the SEI to accompany the goal-driven measurement methodology precisely describes an indicator—including its construction, correct interpretation, and how it can be used to direct data collection and presentation, along with measurement and analysis processes. The indicator template helps an organization to define indicators, or graphical representations of measurement data, which describe the "who, what, where, when, why, and how" for analyzing and collecting measures.

The goal-driven software measurement methodology was provided and implemented in a training course and workshop format to organizations across the spectrum of software/system development and maintenance in government, the DoD, and industry; tailored versions have been offered in industry settings. Participating organizations include the Internal Revenue Service, NASA, Nuclear Regulatory Commission, the U.S. Department of Veterans Affairs, U.S. Army Aviation and Missile Command (AMCOM), Caterpillar, and Xerox.

**The View from Others**

*The GQ(I)M method provides a powerful way for software evaluators to ensure that the software measurement achieves pre-determined business objectives.*

– Andrew Boyd, City University, Department of Information Science, United Kingdom and John A. Boyd, Boyds VI Consulting [Boyd 2002]

*Developing clear and relevant indicators is crucial to measurement success.*

– Terry Vogt, Booz, Allen, Hamilton [Vogt 2008]

### 4.12.3 The Consequence: Successful Measurement Processes That Support an Organization's Business Goals

The goal-driven measurement methodology has proved to be broadly applicable, not just for software measurement. It has been used to establish a system of uniform measures across a global enterprise, assessing the impact of investment in software process improvement and developing a standardization of measurement to reconcile perceived conflicts between what the customer demands and what the "corporate" requires. The goal-driven software measurement process directs attention toward measures of importance rather than measures that are merely convenient.

The goal-driven measurement methodology and the accompanying indicator templates have been used successfully by many organizations in industry and government in diverse settings and with different goals to implement measurement programs. The artifacts developed (such as templates and checklists) and the lessons learned have provided insight to others trying to implement measurement programs. The indicator template that accompanies goal-driven measurement reflects the thinking and practices of multiple organizations over time. It has been shown to reduce cycle time by enabling organizations to leverage their experience and to quickly focus on measurement content rather than form. The indicator template has been adopted and integrated into many organizations' processes. Electronic Data Systems now describes the indicator template as the "cornerstone" of its successful measurement and process improvement effort [Crawford 2004]. In addition, Dr. Rick Hefner of Northrup Grumman has included GQIM in his (Define, Measure,

Analyze, Improve, Control (DMAIC) toolkit for implementation of Six Sigma in the aerospace industry, based on its well-defined approach [Hefner 2011].

### 4.12.4 The SEI Contribution

The "I" in the parentheses distinguishes the SEI-developed GQ(I)M methodology from the closely related GQM methodology introduced and described by Basili and Rombach. In the SEI elaboration of Basili's methodology, an additional intermediate step assists in linking the questions to the measurement data that will be collected. (Experience shows that it is much easier to postulate indicators and then identify the data items needed to construct them than it is to go directly to the measures. Starting with the raw data [measures or data elements] and creating an indicator can lead to convenient or elegant displays that incorporate the data but fail to address the information needed to answer the questions that drove the data collection.)

The GQ(I)M methodology was enhanced with the development of the indicator template that contains fields to precisely document the construction, interpretation, and use of the indicator. It serves as a tactical aid in the execution of the measurement process. It helps to ensure the consistent collection of measures for constructing the indicators and provides a set of criteria for ensuring the consistent interpretation of the measures collected.

### 4.12.5 References

[Basili 1984] Basili, V. & Weiss, D. "A Methodology for Collecting Valid Software Engineering Data." *IEEE Transactions on Software Engineering 10*, 3 (November 1984): 728-738.

[Basili 1988] Basili, Victor R. & Rombach, H. Dieter. "The TAME Project: Towards Improvement-Oriented Software Environments." *IEEE Transactions on Software Engineering 14*, 6 (June 1988): 758-773. IEEE, 1988.

[Boyd 2002] Boyd, A. "The Goals, Questions, Indicators, Measures (GQIM) Approach to the Measurement of Customer Satisfaction with E-Commerce Web Sites." *Aslib Proceedings 54*, 3 (2002): 177-187. MCB UP Ltd.

[Boyd 2010] Boyd, Andrew & Boyd, John A. "Thoughts on Evaluation of SME Strategic Relationships." *International Council for Small Business, 47th World Conference Proceedings.* San Juan, Puerto Rico, June 16-19, 2002. ICSB, 2002. http://sbaer.uca.edu/research/icsb/2002/038.pdf

[Briand 1996] Briand, L.; Differding, C. M.; & Rombach, H. D. "Practical Guidelines for Measurement-Based Process Improvement." *Software Process Improvement and Practices 2*, 4 (December 1996): 253-280.

[Crawford 2004] Crawford, Paul & Stephens, Mark. "Transitioning From Business Objectives to Measurement Objectives." *Proceedings of the European SEPG 2004.* London, England, June 14-17, 2004. European Software Process Improvement Foundation, 2004.

[Hefner 2011] Hefner, Rick. "Process Improvement in the Aerospace Industry: CMMI and Lean Six Sigma." (*Lecture for Computer Science 510, University of Southern California).* 2011. sunset.usc.edu/classes/cs510_2011/ECs_2011/10USC%20Hefner%20CMMI%20LSS.pptx

[Rombach 1989] Rombach, H. Dieter & Ulery, Bradford T. “Improving Software Maintenance Through Measurement.” *Proceedings of the IEEE (PIEEE) 77*, 4 (April 1989): 581-595.

[Vogt 2008] Vogt, Terry. “Goal-Driven Performance Measurement.” *SEER User Conference*, Redondo Beach, CA, April 10, 2008. Golorath Inc., 2008. http://www.galorath.com/index.php/library/userconference2008

# 5 Security

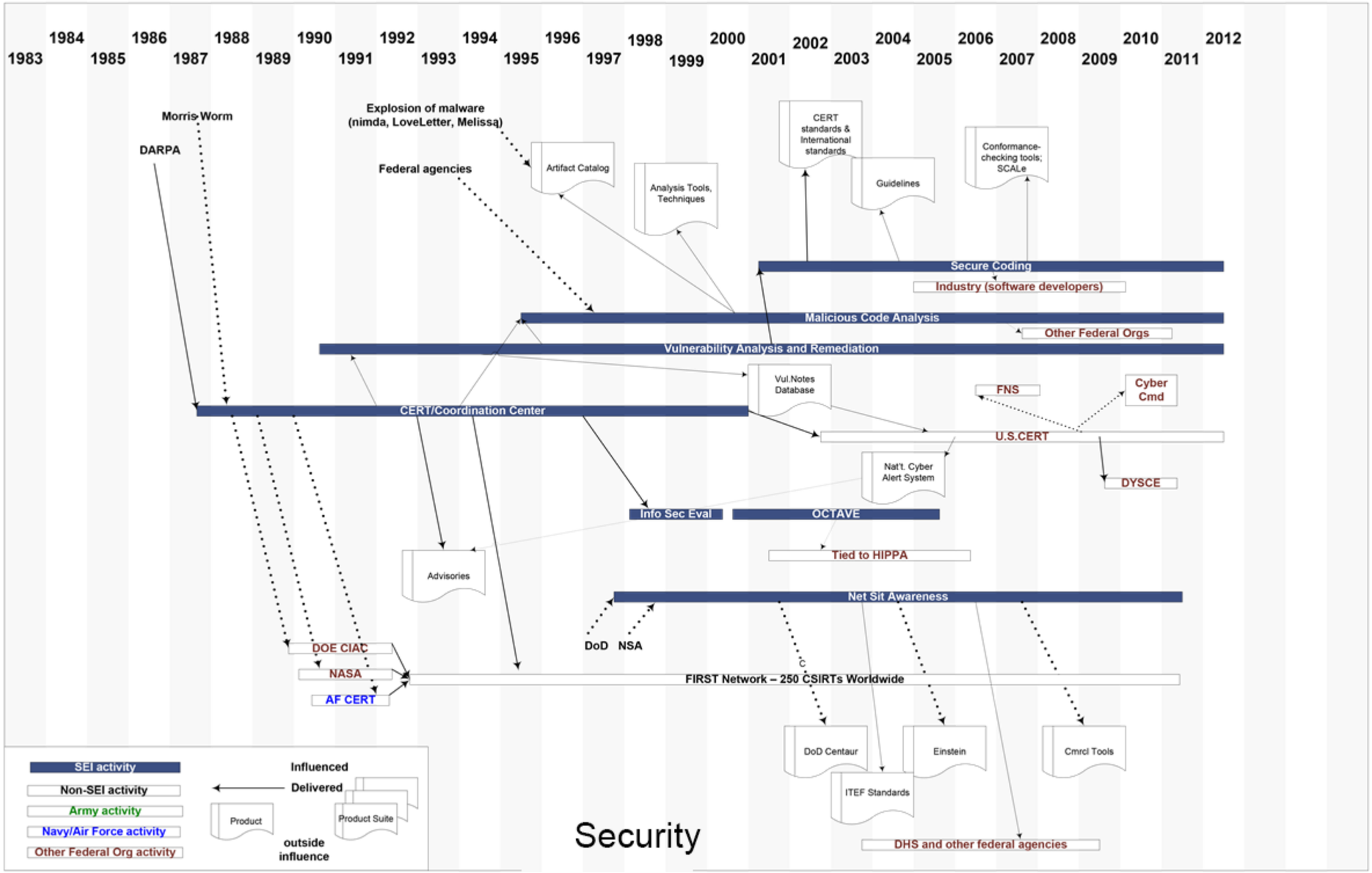


*Figure 6: Security Timeline*

## 5.0 Introduction to Security

In the early days, security was not high on the list of issues facing software engineers, even for those involved in DoD systems, except for those who developed software for classified systems. At best, most used defensive techniques aimed at surviving mistakes users might make. They seldom concerned themselves with the possibility that someone with malicious intent might subvert their systems.

This was particularly true of the internet, which was spawned from the ARPANET [Leiner 2012, Museum 2006]. The ARPANET began as a research-oriented development to offer packet switching as a new paradigm in network construction—it was a research project for researchers. The underlying technology was developed with an open, trusting style. Everyone was expected to be a friendly user. Indeed, there was a culture of cooperation and a willingness to contribute fixes for common problems that was self-regulating.

Initially, the expansion of the ARPANET to the internet simply expanded the user community to a broader segment of the research community. However, once the commercial potential was realized, the rapid growth led to a user profile that more closely reflects the general population. Unfortunately, the general population includes mischievists, thieves, and criminals.

Several people, including those who developed the underlying technology, warned that the internet was not intended to be secure and that there was a serious potential for abuse. Although DARPA began investigating security solutions [FAS 2000], there was little concern among users because there was no "smoking gun"—no indication that anyone would seriously attack the internet or systems on the internet. While there were some attacks, there was still no clear indication of a persistent threat and, therefore, no serious attention given to security.

### 5.0.1 Genesis of the CERT Coordination Center

That ambivalence was rudely shaken on November 2, 1988, when a graduate student released a worm on the internet [ACM 1989]. The Morris Worm (named for its inventor) brought the internet to its knees. For the 72 hours after the release of the worm, the research community, coordinated by two program managers at DARPA, reverse engineered the worm to understand how it functioned, then began to provide advice to systems administrators on removing the worm. Communication was hampered because the worm clogged the network, the primary means of communication for many sites. Moreover, many sites removed themselves from the ARPANET altogether, further hampering communication and the transmission of the solution that would stop the worm. Although the ad hoc collaboration of experts from around the country was effective in defeating the worm, DARPA realized that the worm, though destructive, was reasonably benign in relation to havoc it might have wreaked if Morris had been more malicious. DARPA management knew that there would certainly be more malicious attacks in the future and, the following week, asked the SEI to propose a mechanism that would encourage and support collaboration among technical experts in resolving security problems, and coordinate their response activities in the event of future attacks against the internet and connected systems.

DARPA and the SEI agreed that the SEI would set up a computer emergency response team (now the CERT Coordination Center – CERT/CC) with the following operational concepts:[25]

- a non-government entity, a neutral broker, no regulatory authority
- needing access to key experts
- relying on previous trust relationships
- working with vendors to mitigate vulnerabilities
- building on DARPA's position in the community using the SEI and Carnegie Mellon
- getting agreements in place to bypass bureaucracy
- forming a federation of computer security incident response teams modeled after the CERT/CC

The original concepts have proved to be robust and scalable, and many of the early relationships have endured.

Over the next few weeks, the CERT charter was hammered out. These are the terms of the charter:

> CERT is chartered to work with the internet community in detecting and resolving computer security incidents, as well as taking steps to prevent future incidents. In particular, our mission is to
>
> - Provide a reliable, trusted, 24-hour, single point of contact for emergencies.
> - Facilitate communication among experts working to solve security problems.
> - Serve as a central point for identifying and correcting vulnerabilities in computer systems.
> - Maintain close ties with research activities and conduct research to improve the security of existing systems.
> - Initiate proactive measures to increase awareness and understanding of information security and computer security issues throughout the community of network users and service providers.
> - Serve as a model for other incident response organizations.

The CERT/CC began operating on December 6, 1988, with one SEI staff member and one administrative assistant, along with three part-time members "borrowed" from the IT department. Just hours after DARPA announced the CERT/CC in a press release on December 12, 1988, the center received its first hotline call reporting a security incident, and it released its first advisory before the year's end. The CERT/CC has never since been without an active incident; the activity only increased as time went on [Howard 1997, Moitra 2004].

25 Private recollections of Larry Druffel and Bill Scherlis regarding an agreement between Druffel and the DARPA deputy director. Many people contributed to the discussions behind the operational concept, from the Livermore Labs response team, Defense Systems Information Agency (DISA), National Computer Security Center (NCSC), and National Institute of Standards and Technology (NIST), and from law enforcement agencies such as the Department of Justice, the Federal Bureau of Investigation, the U.S. Secret Service, and others.

The CERT/CC helps guard against devastating consequences from incidents by sharing its expertise with government and other incident handlers and their managers, including US-CERT. The CERT/CC protects the U.S. information infrastructure by providing technical assistance in repairing compromised systems and limiting the damage caused by high-impact attacks. Access to composite data enables the U.S. Department of Defense (DoD) and federal agencies to have a comprehensive view of attack methods, vulnerabilities, and the impact of attacks.

### 5.0.2 Evolution of the CERT Division

In the years following CERT/CC's establishment, the DoD and federal agencies became highly dependent on the internet, as did businesses, including critical infrastructure providers. Moreover, they moved away from proprietary software to adopt common information technologies. To better manage these changes, the CERT/CC began addressing a wider range of security issues, and the larger CERT program was formed, later becoming a division. Through the CERT Division, the SEI develops and promotes the use of appropriate technology and systems management practices to resist attacks on networked systems, to limit damage, and to ensure continuity of critical services.

### 5.0.3 Range of Issues

Establishment of the CERT/CC was the SEI's introduction to a broad range of software and network security issues.

The first issue, in accord with the CERT charter, was incident response (IR). From the beginning it was clear that the need for skilled incident responders and response organizations would grow. In addition to serving as a key response organization, the CERT/CC took on the task of developing the mentoring and training programs, training delivery platforms, and cyber exercise platforms that would scale to meet the growing workforce development need. After defining best practices and sharing them with the IR community, the CERT/CC worked at the organizational level then moved to the national level and its special technical IR needs; these activities included helping to establish and providing ongoing support to US-CERT. Recognizing the need for a network of incident responders, the CERT/CC was one of the founding members of the Forum of Incident Response and Security Teams (FIRST). Later, GFIRST was formed to meet the particular needs of government CSIRTs (computer security incident response teams). In effect, the SEI not only spawned this international collection of cooperating organizations, it has also been a leader in coordinating technical support to evolving incidents throughout the world.

Perpetrators of incidents take advantage of vulnerabilities in software products, so another early issue was vulnerability analysis, also called for in the CERT charter. This work began with collecting and categorizing vulnerability reports and establishing working relationships with more than 600 vendors to mitigate security problems responsibly. The CERT/CC then established an initiative to develop vulnerability discovery tools and analysis techniques, then provide them to vendors, leading to fewer vulnerabilities in released software as the vendors begin using them in their software development process. As a result, DoD and others' acquisition teams are assured of more secure software products out of the box and increased resistance to attacks.

Collecting and analyzing malicious code was a logical next step as vulnerability analysts and incident responders saw "malcode" and "malware" toolkits exploiting vulnerabilities. The CERT/CC

collects over one million pieces of malicious code each month, entering the malware into its Artifact Catalog. As incident responders deal with the growing frequency of malcode-based attacks, they need the ability to analyze malicious coded packages quickly, determine the effects of the malicious code, and understand how to mitigate those effects. Similarly, law enforcement agents investigating cybercrimes need the ability to identify the source of malicious code attacks. The CERT/CC has an ongoing effort to develop analysis techniques, tools, and the training to help other responders and investigators increase their capability to research and mitigate malicious code-based attacks. Automated tools significantly decrease analysis time and enable researchers, analysts, and investigators to be increasingly effective at identifying and understanding malicious code. As a result of CERT/CC work, federal agencies have recovered from serious cyber attacks quickly and solved cybercrimes.

The CERT Secure Coding Initiative grew from the conviction that it is not sufficient to merely respond to security compromises and the vulnerabilities behind them. Rather, vendors need to release less vulnerable software in the first place. CERT secure coding guidelines and standards help developers prevent vulnerabilities by addressing them early in product development. Tools for analyzing the code enable vendors to validate conformance to the standards. The result is more secure out-of-the-box software products and protection for DoD, federal agency, and business systems. Secure coding practices are in use by virtually all defense contractors as well as industry vendors.

DoD needs led to the development of network situational awareness tools, along with analysis techniques for quantitatively characterizing threats and targeted intruder activity. At the time, DoD security operations were driven by known issues that were dealt with in real time without knowledge of the threat behind them. The DoD needed retrospective analysis with historical data and a baseline of the Non-Secure Internet Protocol Router Network (NIPRnet) as a whole. CERT toolsets are now in use at large operations centers in the DoD and the Department of Homeland Security; the tools collect and analyze large volumes of data that enable analysts to understand broad network activity and take appropriate action. US-CERT has used Einstein[26] to meet statutory and administrative requirements of DHS to help protect federal computer networks and the delivery of essential government services. The CERT Division gains far-reaching influence with open source tools and participation in Internet Engineering Task Force (IETF) working groups.

DoD interest in incidents by insiders—staff, former staff, and contractors—prompted the CERT Division's initial work on insider threat. CERT analysts realized just how critically serious insider threat is when they met with the Olympic Committee on cyber aspects of the Salt Lake City Olympic Games, a U.S. Secret Service National Special Security Event (NSSE). It formed a CERT insider threat group, which collected hundreds of case studies of actual incidents and worked with federal law enforcement profilers to examine both the technical and behavioral aspects. This research has expanded to specific domains and types of attack, including espionage, enabling the recently established CERT Insider Threat Center to provide more specific and actionable guidance. Using the center's results, information assurance staff and counterintelligence analysts have implemented technical controls for catching insiders. The DoD and federal civilian agencies are identifying insider threat proactively using SEI techniques and tools instead turning to forensics after a crime.

26 Einstein is an intrusion detection system that monitors the network gateways of government departments and agencies in the United States for unauthorized traffic (en.wikipedia.org/wiki/Einstein).

The SEI concern about risks posed to national and economic security provided the impetus for evaluation methods that give the DoD and other organizations a view of risk to their information systems, along with practices for protecting those systems. OCTAVE is a suite of techniques, methods, and tools for assessing information security risks to critical government and business assets, the foundation for planning continuous risk management. A related assessment is Computer Network Defense (CND) metrics, which focus on risks in CSIRT incident management. The U.S. Army and Air Force use OCTAVE to help meet HIPPA regulations, and the technique has been taught at DoD medical treatment facilities around the globe.

As part of its effort to influence product development, the SEI began looking at how to incorporate security early in the software development lifecycle. The Cybersecurity Engineering group focuses on using engineering solutions to address this challenge. The DoD and federal agencies benefit from frameworks and methods that support decisions from acquisition through operation, which organize research and practice areas for building assured systems, and that guide measurement and analysis. SQUARE (Security Requirements Engineering) gives developers a process for identifying security and privacy requirements from the start; A-SQUARE, an addition to the SQUARE suite, aids in acquisition of stable products with security as an integral attribute rather than an add-on.

Future research and development will enable the CERT Division to keep up with changing technology, risks, attacks, and DoD software assurance needs.

### 5.0.4 References

[ACM 1989] *Communications of the ACM 32,* 6 (June 1989): Entire issue devoted to the subject of the Morris Worm. Example article: Eugene H. Spafford, "The Internet Worm: Crisis and Aftermath:" 678-687.

[FAS 2000] Federation of American Scientists. "Blacker." http://www.fas.org/irp/program/security/blacker.htm (2000).

[Howard 1997] Howard, John D. "An Analysis of Security Incidents on the Internet 1989-1995.", PhD diss., Carnegie Mellon University, 1997. https://resources.sei.cmu.edu/library/asset-view.cfm?assetID=52454

[Leiner 2012] Leiner, Barry M.; Cerf, Vinton G.; Clark, David D.; Kahn, Robert R., et al. "Brief History of the Internet." Internet Society. http://www.internetsociety.org/internet/internet-51/history-internet/brief-history-internet (2012).

[Moitra 2004] Moitra, Soumyo & Konda, Suresh. "An Empirical Investigation of Network Attacks on Computer Systems," *Computers and Security 23*, 1 (February 2004): 43-51.

[Museum 2006] Computer History Museum. "Internet History." http://www.computerhistory.org/internet_history (2006).

## 5.1 CERT Coordination Center

### The Challenge: Responding to Internet Security Incidents

In 1988, the ARPANET, a network set up primarily for academic and government researchers, had its first automated network security incident, usually referred to as the Morris Worm [Goff 1999]. A student at Cornell University, Robert T. Morris, wrote a program that exploited several vulnerabilities to copy itself and run on a second computer, with both the original code and the copy, repeating these actions in an infinite loop to other computers on the ARPANET. This "self-replicating automated network attack tool" caused a geometric explosion of copies to be started at computers all around the ARPANET. The worm used so many system resources that the attacked computers (10 percent of the network) could no longer function [Marsan 2008].

By that time, the ARPANET had grown to more than 88,000 computers and was the primary means of communication among network security experts. With the ARPANET effectively down, it was difficult to coordinate a response to the worm. Many sites removed themselves from the ARPANET altogether, further hampering communication and the transmission of the solution that would stop the worm.

Now, with nearly a billion hosts on the internet (July 2013), the potential impact of incidents can be worldwide. Closer to home, there are 2,305,000 government (.gov) hosts and 2,592,000 military hosts.[27] With businesses, including critical infrastructures relying on computer systems, the risk to the United States is great. It is essential to respond to incidents quickly and minimize the damage [Kaplan 2011, Gupta 2010a, 2010b].

### 5.1.1 A Solution: Coordinating Incident Response

The 1988 worm was a wake-up call. Following a series of meetings in Washington, DC, the DARPA asked the SEI to immediately create a mechanism to give security experts a central point for coordinating responses to security incidents and to help prevent incidents. The SEI worked with DARPA program managers to develop the concept of the CERT Coordination Center. DARPA agreed to fund the CERT/CC, charging the newly formed center with serving as a central point for identifying and correcting vulnerabilities in computer systems, keeping close ties with research activities and conducting research to improve security, and initiating "proactive measures to increase awareness and understanding of information security and computer security issues throughout the community of network users and service providers" [Fithen 1994].[28]

The CERT/CC received its first hotline call reporting a security incident just hours after DARPA issued a press release announcing the center, and it published the first CERT advisory within a month. The activity only increased as time went on [Howard 1997, Moitra 2004], as did the potential damage from incidents; see, for example, articles by Hulme and by Stilgherrian [Hulme 2011, Stilgherrian 2011]. Recognizing the need for a network of incident responders, the CERT/CC was one of the founding members of the Forum of Incident Response and Security

27 Internet Systems Consortium (ICS) Domain Survey (http://www.isc.org/services/survey).

28 Pressclips, a collection of articles published after the CERT/CC was formed, described the center and evidence of the need for it.

Teams (FIRST), which was formed in 1989 and now boasts 289 members worldwide (as of January 2013).[29]

In the early years, the CERT/CC staff responded to every incident report and worked closely with individuals reporting incidents. This activity enabled the staff to understand the practices involved in incident response and determine how to make them repeatable. Upon noticing that many people made the same mistakes, the center began writing "tech tips" and checklists (an early tech tip was a UNIX security checklist). By the mid-1990s, the CERT/CC had accumulated enough knowledge and experience to codify processes and teach others how to do incident response, resulting in the first training courses and a handbook for CSIRTs [West-Brown 2003].[30] The CERT/CC also assisted in the establishment of response teams; for example, CERT experts helped the Army with structure, organizational listing, and training for ACERT, the Army's incident response team. After working at the organization level, the CERT/CC moved to the national level and the nation's special technical needs. The center played a significant role in the creation and continued evolution of US-CERT, the national CSIRT for the United States, and Q-CERT, the national CSIRT of Qatar. As industry capacity grew, the CERT/CC focused more on codifying best practices and growing capacity.

**The View from Others**

*Since CERT was formed it has been a great help to me and my several employers since that time. I wish to thank you for your great work!*

– a physicist working in a government institute of science

*Thanks to all of you – you're doing a great service to the information security community. Keep up the good work!*

– an information security officer

*These online docs were very useful. In fact the checklist was how we found the network sniffer….*

– a user of the Intruder Detection Checklist

CERT/CC staff now helps loosely coordinate national CSIRTS, supporting central points such as Information Sharing and Analysis Centers (ISACs), and providing operational coordination for critical infrastructure/key resource (CIKR) led by the DHS and DoD. The CERT/CC holds an annual technical meeting that helps it effectively share knowledge and tools. Its location at the SEI and Carnegie Mellon enables it to serve as a neutral, trusted, third party to coordinate responses to high-impact incidents across geographic, national, political, and economic boundaries. The CERT/CC is concentrating on threats that affect national and economic security, with a focus on government and critical infrastructure and on threats from the most serious adversaries, especially threats that do not yet have a commercial solution. The center seeks ways to identify threats and remediate them, concentrating on the technological cutting edge.

29 Information and a list of members are available at http://www.first.org.

30 Additional materials for CSIRTS are at http://www.cert.org/incident-management/csirt-development/index.cfm.

### 5.1.2 The Consequence: Knowledgeable Incident Responders, Coordinated Response

The DoD and federal agencies are highly dependent on the internet, as are businesses, including critical infrastructure operators. They can better guard against devastating consequences from incidents because the CERT/CC shares its expertise with government and other incident handlers and their managers. The U.S. information infrastructure is better protected because of access to CERT/CC technical assistance in repairing compromised systems and limiting the damage caused by high-impact attacks. A community of incident response practitioners is increasingly effective at improving internet-connected systems' resistance to attack as well as detecting and resolving successful attacks on those systems.

### 5.1.3 The SEI Contribution

Before the CERT Coordination Center was established, there were many security experts across the country, including those who reverse engineered the Morris Worm and found a way to stop it. The SEI contribution was to establish secure channels for sharing information and coordinating response in the face of incidents that threaten networks and the information they carry. The CERT/CC continues to reach out to existing experts and also to help others gain expertise. It raises awareness in the government and elsewhere of security risks and the need to be prepared to respond quickly and effectively to potentially devastating security breaches. The center was instrumental in forming a global network of incident responders, facilitating response to incidents that cross national, political, and geographic boundaries.

### 5.1.4 References

[DARPA 1988] Defense Advanced Research Projects Agency, Press release, December 13, 1988.

[Fithen 1994] Fithen, Katherine & Fraser, Barbara. "CERT Incident Response and the Internet." *Communications of the ACM 37*, 8: 108-113.

[Goff 1999] Goff, Leslie. "Technology Flashback: Worm Disables Net." *Computerworld* (October 4, 1999): 78.

[Gupta 2010a] Gupta, Upsana. "Incident Response: Drafting the Team." *Banking Info Security*. November 5, 2010. http://www.bankinfosecurity.com/articles.php?art_id=3060

[Gupta 2010b] Gupta, Upsana. "Incident Response: 5 Critical Skills." *Banking Info Security*. November 4, 2010. http://www.bankinfosecurity.com/articles.php?art_id=4214&pg=1

[Howard 1997] Howard, John D. "An Analysis of Security Incidents on the Internet 1989-1995." PhD diss., Carnegie Mellon University, 1997.

[Hulme 2011] Hulme, George V. "Medical Data Breaches Soar, According to Study." *CSO*. December 2, 2011. http://www.cso.com.au/article/409040/medical_data_breaches_soar_according_study

[Kaplan 2011] Kaplan, Dan. "Major Breach: Ground Control." *SC Magazine*. September 8, 2011. http://www.scmagazine.com.au/Feature/271380,major-breach-ground-control.aspx

[Killcrece 2008] Killcrece, Georgia; Zajicek, Mark; & Ruefle, Robin. “Creating and Managing Computer Security Incident Response Teams (CSIRTS)” (tutorial). 20th Annual FIRST Conference, June 20-27 2008, Vancouver, Canada.

[Marsan 2008] Marsan, Carolyn Duffy. “Morris Worm Turns 20: Look What It’s Done.” *Network World*. October 30, 2008. http://www.networkworld.com/news/2008/103008-morris-worm.html

[Moitra 2004] Moitra, Soumyo & Konda, Suresh. “An Empirical Investigation of Network Attacks on Computer Systems.” *Computers and Security 23*, 1: 43-51.

[Stilgherrian 2011] Stilgherrian. “LulzSec, WikiLeaks, Murdock: Hacking’s Fourth Wave.” *CSO Online.* August 8, 2011. http://www.cso.com.au/article/396368/lulzsec_wikileaks_murdoch_hacking_fourth_wave/#closeme

[West-Brown 2003] West Brown, Moira; Stikvoort, Don; Kossakowski, Klaus-Peter; Killcrece, Georgia; Ruefle, Robin; & Zajicek, Mark. *Handbook for Computer Security Incident Response Teams (CSIRTs)* (CMU/SEI-2003-HB-002). Software Engineering Institute, Carnegie Mellon University, 2003. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=6305

## 5.2 Vulnerability Analysis and Remediation

### 5.2.1 The Challenge: Software Vulnerabilities

In producing software products, engineers unintentionally create vulnerabilities. When the problems are discovered before a compromise—perhaps by researchers or the vendors themselves—organizations and individuals have a chance to protect their systems from attack. When vulnerabilities are discovered by people with malicious intent, the result can vary from inconvenient to disastrous [CIOinsight 2011]. Attackers, who once worked alone, share techniques and tools [Pollak 1998], and they are quick to adapt their tools to exploit newly disclosed vulnerabilities. Some attacks are used in business or warfare, for raising money and for undermining competitors/adversaries [Shimeall 2001].

The DoD, federal agencies, and physical critical infrastructures have become increasingly exposed to software vulnerabilities, and it is increasingly important for them to respond quickly. Every organization that evaluates vulnerability reports must invest valuable time, often duplicating others' efforts; failure to respond to vulnerabilities in an appropriate, timely manner puts the organization at greater risk. Comprehensive configuration control and patch management remains a challenge for large enterprises. The problem has international aspects: software is produced all over the world and used in the U.S., and compromised computers outside the U.S. present risks to the U.S. infrastructure.

### 5.2.2 A Solution: Vulnerability Analysis, Remediation, and Discovery

The CERT/CC charter includes the charge to serve as a central point for identifying and correcting vulnerabilities in computer systems. Vulnerability analysis goes hand-in-hand with incident handling since understanding the vulnerability may guide the recovery process and prevent future compromises. Soon after the CERT/CC began operations at the SEI, it set up secure channels for reporting vulnerabilities, and it formed trusted working relationships with vendors in case a vulnerability is discovered. It raised awareness of software development practices to improve and encouraged releasing software with more secure default configurations. As a neutral third party, the CERT/CC could report vulnerabilities to vendors without revealing the identity of the reporter and could work with competing vendors whose products contain the same vulnerability. In 2012, the CERT/CC had trusted interactions with more than 600 technology vendors. System administrators and users benefit. They need to know if patches are available and to what extent they address the vulnerability. If patches are not available, they need interim strategies that help mitigate the impact on vulnerable systems. The CERT/CC has always taken care to disclose information about security weaknesses in a way that minimizes the chances of subsequent compromise through that flaw, releasing information that balances the community's need to protect systems and the vendors' need for time to develop and test solutions.

The CERT/CC wrote a vulnerability disclosure policy that considers these equities. This policy provides a vendor 45 days to remediate a problem prior to public disclosure. Vendors can request more time before public disclosure if a particular vulnerability is complicated to fix. However, if there is active exploitation of a vulnerability, it will be disclosed. Since 2000, the CERT/CC has facilitated mitigation of vulnerabilities and disseminated the information through the publication of products called *vulnerability notes*.

The CERT/CC built on its vast collection of data to create a knowledgebase[31] that addresses the challenges of how to structure, distribute, and maintain security incident and vulnerability information in a useful form. Work on the knowledgebase began in 1996 with funding from the Air Force Materiel Command, Rome Laboratory, to build a prototype. The resulting work captured the interest of additional sponsors, and the Air Force Information Warfare Center sent associates to the SEI to work with CERT/CC staff to gain expertise to take back to their organizations.

After chronicling the same implementation flaws repeatedly for more than 15 years, the CERT experts began developing vulnerability discovery tools to help reduce the number of vulnerabilities in software before it is deployed. Starting with ActiveX, they developed the Dranzer tool and released it to vendors in 2006 and to the open source community in 2009. They also published a paper describing the history, motivations, and rationale for Dranzer, along with early results [Dormann 2008]. In 2010, they released the Basic Fuzzing Framework (BFF) to help developers and testers apply effective black-box fuzz testing to their software. Another tool, Failure Observation Engine (FOE), performs on Windows systems the same functions as the BFF. Additionally, CERT triage tools assist software vendors and analysts in identifying the impact of defects discovered through techniques such as fuzz testing. The CERT/CC continues to develop and test tools on current software. The goal is to provide vendors with user-friendly, efficient tools and techniques they can incorporate into their development process and prevent vulnerabilities before release.

**The View from Others**

(These comments are from users of CERT publications and knowledgebase; identities are protected.)

*Thanks for the heads up.....you people are the greatest...in particular, this last one sewed up a hole that was literally a breach in front lines for our team, so speak.*

– a system administrator

*Thank you for providing a wonderful mechanism for tracking and notification concerning system vulnerabilities.*

– a corporate webmaster

*We have seen several potential defects revealed using Dranzer. It is certainly a useful tool, well documented, and really easy for an engineer to use.*

– a technical staff member of a large technology vendor

### 5.2.3 The Consequence: Improved Vendor Practices, Well-Informed System Mangers

Vendors' practices have improved, resulting in the improved security of their products. With CERT/CC influence, vendors have not only improved their development practices, but they also provide safer default configurations and free, broad distribution of security updates. Also, the use of vulnerability discovery tools is leading to fewer vulnerabilities in released software. Several software vendors have told CERT staff privately that they either have or plan to incorporate Dranzer and/or BFF into their software development practices; one company's job description noted a preference for those with Dranzer experience.

31 For a brief description, see http://www.cert.org/vulnerability-analysis/knowledgebase/index.cfm.

System and network administrators and managers are now better informed. They use the data CERT/CC provides from its incident and vulnerability work to prioritize their security improvement efforts and focus on areas that are known to be vulnerable to attack. With this data, they are aware of the current state of information security products, system management and security practices, and intruder methods. The U.S. government's access to the composite data enables the DoD and federal agencies to have a comprehensive view of attack methods, vulnerabilities, and the impact of attacks on information systems and networks.

### 5.2.4 The SEI Contribution

The SEI, through its CERT/CC, filled the gap in the 1990s, when major vendors released little vulnerability information. Previously, there was some uncoordinated work on vulnerabilities by vendors and researchers; and system and network administrators sometimes worked with vulnerabilities on their own, primarily in response to exploitation. The CERT/CC connected vulnerability researchers to vendors and reduced duplicate efforts by individual groups. The CERT/CC's neutrality and credibility gave both vulnerability reporters and vendors confidence that the information they provided was treated with discretion. The center's position also enabled it to coordinate vulnerabilities affecting products of multiple vendors. Importantly, it pioneered "responsible disclosure."

The CERT/CC's early interactions with technology vendors helped to increase vendor sensitivity to security requirements and improve the security of their products, including basic, as-shipped products. In addition, vendors originally intended to charge users for security updates; the CERT/CC was instrumental in vendors' current practice of providing free updates and issuing bulletins. Vendors now issue their own bulletins, and information sources such as the National Vulnerability Database (NVD)[32] are available. Thus, the CERT/CC can concentrate on vulnerabilities with critical impact on the U.S. economy, information infrastructure, and national security and can focus on techniques and tools to enhance vulnerability discovery.

### 5.2.5 References

[CIOinsight 2011] CIOinsight. "UBS Rogue Trader: An Enterprise Security Wake-Up Call." *CIOinsight,* September 16, 2011. http://www.cioinsight.com/c/a/Latest-News/UBS-Rogue-Trader-Underscores-Insider-Threats-Facing-Enterprises-368962/?kc=rss

[Dormann 2008] Dormann, Will & Plakosh, Dan. "Vulnerability Detection in ActiveX Controls through Automatic Fuzz Testing." Software Engineering Institute, Carnegie Mellon University, 2008. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=53466

[Pollak 1998] Pollak, Bill. "Interview with Rich Pethia," *news@sei.* Software Engineering Institute, Carnegie Mellon University, 1998. http://www.sei.cmu.edu/library/abstracts/news-at-sei/spotlightdec98.cfm

[Shimeall 2001] Shimeall, Timothy, Williams, Phil, & Dunlevy, Casey. "Countering Cyber War." *NATO Review 49*, 4 (Winter 2001): 16-18.

32 The NVD can be found at nvd.nist.gov.

## 5.3 Malicious Code Analysis

### 5.3.1 The Challenge: Malicious Code

Malicious code,[33] or malware, has affected the DoD, including the Pentagon and federal agencies, for well over a decade [Brewin 1999, Verton 1999, Ruppe 2001]. Retail businesses, banks, and a stock exchange have all been victims. Attackers have developed hundreds of thousands of pieces of malicious code; some well-publicized early ones were known as Melissa, LoveLetter, Nimda, and Code Red. Viruses have affected an estimated two-thirds of Fortune 500 companies and cost victims billions of dollars in productivity [Frontline 2000].

Malicious code exploits software vulnerabilities. However, in the 1990s, the community had not matured to the point of sharing tools and techniques, and doing cross mentoring. Only a few people had strong skills in reverse engineering malicious code to understand how it works; and rather than work together, they seemed to compete.

### 5.3.2 A Solution: Malicious Code Database and Analysis

In the late 1990s, the CERT/CC at the SEI began studying the malicious software that got into systems through vulnerabilities and social engineering—tricking people into actions that allowed malware onto their computers. The CERT/CC security experts had already become skilled at analyzing software vulnerabilities. The malicious code work was prompted by CERT/CC incident responders' seeing the role malware played in computer security incidents. By 2000, malicious code was exploding, and the staff began more thorough analysis efforts [Bair 1999]. In dissecting malware, they compiled a wealth of information not only about malicious code but also about how technology fails, what assets adversaries target, how they acquire targets, and who the adversary is.

The CERT/CC began building a database, which became known as the Artifact Catalog,[34] a repository of malicious code and related analysis that informs triage decisions and provides a basis for cross-threat analysis. The malware analysts also worked to establish relationships within the security community: for example, by raising awareness at professional conferences and by working with experts through email. This visibility led to government interest and funding for the SEI malicious code work. Congress had become interested in solving the problem of malicious code and heard testimony on the subject. One of those testifying was tasked with "doing something" regarding malware, and the CERT Artifact Catalog proved to be the answer to that charge. Funding subsequently followed, allowing the work to grow.

The CERT/CC created a separate, focused malicious code team, which had these primary goals: (1) improve approaches to reverse engineering, (2) bring those doing reverse engineering together into a community that would work collaboratively and learn from each other, and (3) reduce duplicate work in the community to make effective and efficient use of limited resources, as well as

33 Malicious code is a form of cyber "tradecraft" used by adversaries to subvert the security posture and compromise the assets of organizations.

34 The original intent of the catalog was to collect things left behind on an attacked machine, which the staff referred to as *artifacts.* These artifacts included malicious code but also logs and other files or items left behind by an intruder.

understand the relationship among the various groups in the community. They defined the following plan of work and are implementing it successfully.

1. Bring reverse engineers together in workshops, training, and mentoring sessions to allow them to learn from each other and to increase the skills and capabilities of reverse engineers with less experience. The team held its first invitational Malicious Code Workshop in October 2004 to discuss the top challenges in malware. By 2012, they had held seven Malicious Code Workshops and seven Malicious Code Training Workshops, the latter focusing on sharing analysis techniques.
2. Build a set of tools and processes to collect malicious code and make that collection available to collaborators so that the number of collection infrastructures can be minimized, particularly within the U.S. government. A catalog of malicious code and analysis processes was developed and is available to malware analysts and researchers. Development and analysis continue. The limited-access Artifact Catalog contained more than 80 million files in January 2013, with typically a million new samples ingested weekly.
3. Automate as many analysis functions as possible so limited resources can be spent on only the novel and the most important malware samples. Reverse engineering is a time-consuming technique—an analyst dissects every instruction in the malware. The SEI malware experts have codified some of their reverse engineering expertise into automated tools, which they share with other analysts in the DoD and intelligence community. The SEI analysts use the tools themselves for research and to fulfill requests from government agencies. The team refines the tools and develops additional ones in the process [Householder 2011; Cohen 2009, 2010].

### 5.3.3 The Consequence: Faster Response to Malicious Code Attacks, Better Control

Malware analysts at government operation centers have advanced tools, shortening their response times; they also have specialized analytical support from the SEI through the CERT/CC artifact analysis team. The Artifact Catalog provides them with an extensive collection of malicious code that supports trending and related research. The automated tools significantly decrease analysis time and enable analysts and researchers to be increasingly effective at identifying and understanding malicious code—understanding that is essential to gaining control over attacks and limiting the damage they cause. The result is increased safety for U.S. government and agency systems.

By providing analysis and serving as expert witnesses [Poulsen 2008, Ove 2010, Cruz 2011], SEI malicious code experts have helped to shut down and apprehend a major identity theft and fraud ring, whose activities caused more than $4 million in losses, and helped convict a perpetrator of wire fraud that cost financial institutions an estimated $86 million [Mills 2009].

### 5.3.4 The SEI Contribution

The SEI Malicious Code team has collaborative relationships with both the DoD and the intelligence community. Because of the CERT/CC malware work, SEI collaborators and sponsors have the tools and education needed to streamline analysis and more quickly answer specific questions. The SEI has been a resource to build new capabilities in the U.S. government. Its malware experts

have put advanced tools in the hands of malware analysts at government operation centers, shortening their response times, and have also provided specialized analytical support to federal agencies. It fosters research with the Artifact Catalog and serves as a neutral third party for collecting, categorizing, and analyzing malware.

The SEI has reached out to the community, bringing disparate groups together to collaborate on identifying significant issues and sharing tools and techniques. The Malicious Code Workshops give experts a forum for sharing information and techniques and also help build a community. The SEI malware experts have reached more than 1,100 people through the Malicious Code Workshops and have reached even more through professional conferences, meetings with individual government agencies/departments, and email.

### 5.3.5 References


[Bair 1999] Bair, Jeffrey, Associated Press. "Virus Fighters Eye How, Not Who." *The Indiana Gazette*, (April 9, 1999): 9. Archived at http://www.newspaperarchive.com/SiteMap/FreePdfPreview.aspx?img=112050250

[Brewin 1999] Brewin, Bob & Verton, Daniel. "Melissa Tests DoD Procedures." *Federal Computer Week*, 1999. https://fcw.com/articles/1999/04/11/melissa-tests-dod-procedures.aspx

[Cohen 2009] Cohen, C. & Havrilla J. "Malware Clustering Based on Entry Points." *CERT Research Annual Report 2008.* Software Engineering Institute, Carnegie Mellon University, 2009. http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=54240

[Cohen 2010] Cohen, C. & Havrilla J. "Function Hashing for Malicious Code Analysis," *CERT Research Annual Report 2009.* Software Engineering Institute, Carnegie Mellon University, 2010. http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=51314

[Cruz 2011] Cruz, Natalie. "Internet Hacker Bruce Raisley Gets Prison for Creating Botnet Computer Virus." *Newsroom New Jersey*, April 17, 2011. http://www.newjerseynewsroom.com/science-updates/internet-hacker-bruce-raisley-gets-prison-for-creating-botnet-computer-virus (2011).

[Frontline 2000] Power, Richard. "The Financial Costs of Computer Crime." Interviews with security experts. http://www.pbs.org/wgbh/pages/frontline/shows/hackers/risks/cost.html (2011).

[Householder 2011] Householder, Alan. "Announcing the CERT Basic Fuzzing Framework 2.0." *CERT/CC Blog* February 28, 2011. http://www.cert.org/blogs/certcc/post.cfm?EntryID=66

[Mills 2009] Mills, Elinor. "'Iceman' Pleads Guilty in Credit Card Theft Case." *CNET*. June 29, 2009. http://news.cnet.com/8301-1009_3-10275442-83.html (2009).

[Ove 2010] Ove, Torsten. "Prolific Computer Hacker Gets 13 Years in Prison." *Pittsburgh Post-Gazette* (February 12, 2010). http://www.post-gazette.com/nation/2010/02/12/Prolific-computer-hacker-gets-13-years-in-prison/stories/201002120174

[Poulsen 2008] Poulsen, Kevin. "Feds Charge 11 in Breaches at TJ Maxx, OfficeMax, DSW, Others." *Wired.* August, 5, 2008. http://www.wired.com/threatlevel/2008/08/11-charged-in-m/

[Ruppe 2001] Ruppe, David. "Hit by Virus, Pentagon Web Site Access Blocked." *ABC News.* July 23, 2001. http://abcnews.go.com/International/story?id=80758&page=1#.Tw8C0vKyBac

[Verton 1999] Verton, Daniel. “Melissa Takes Down Marine Corps E-mail.” *Federal Computer Week.* March 31, 1999. http://www.cnn.com/TECH/computing/9903/31/melissamarine.idg/index.html (1999).

## 5.4 Secure Coding

### 5.4.1 The Challenge: Preventing Software Vulnerabilities

Software vulnerabilities open the Department of Defense, other federal agencies, and businesses to attacks that could compromise their systems' integrity or expose or modify their critical information. Software vulnerabilities also put our nation's critical infrastructure at risk. Successful exploitation of these vulnerabilities has severe consequences: financial loss, loss or compromise of sensitive data, damage to critical systems, and loss of productivity.

The traditional, reactive approach of mitigating software vulnerabilities after the product's release is expensive and leaves software users exposed and, frequently, compromised until a patch is released—if customers can keep up with patches at all. Some vulnerabilities are never patched. Preventing the introduction of software vulnerabilities during software development is a proactive, efficient way to reduce risk before the software is ever deployed.

**The View from Others**

*We are thrilled to be the first company to deliver a CERT C compliant programming checker as we believe this new standard will play a significant role in the development of higher quality systems that are more robust and more resistant to attack.*

– Ian Hennell, LDRA Operations Director [Businesswire 2008]

*I'm an enthusiastic supporter of the CERT Secure Coding Initiative. Programmers have lots of sources of advice on correctness, clarity, maintainability, performance, and even safety. Advice on how specific language features affect security has been missing. The CERT® C Secure Coding Standard fills this need.*

– Randy Meyers, Chairman of ANSI C [Seacord 2013]

### 5.4.2 A Solution: Secure Coding Standards and Practices

The CERT/CC has analyzed and cataloged thousands of software vulnerabilities and discovered that many share the same common errors. Deficient or error-prone constructs in the programming languages were frequently a factor. In 2003, the SEI formed the Secure Coding Initiative, whose goals were to enumerate errors in coding that can result in software vulnerabilities and to develop and promote mitigation strategies.[35] By engaging more than a thousand security researchers, language experts, and software developers, the initiative produced secure coding standards for common software development languages such as C and Java. These standards guide programmers to avoid coding errors that lead to vulnerabilities; the standards also provide solution examples. Having standards encourages programmers to follow uniform coding rules and guidelines determined by the requirements of a project or organization, rather than by personal coding preferences or familiarity.

35 Details about the work of the Secure Coding Initiative can be found at http://www.cert.org/secure-coding.

Secure coding standards give developers a known good model to which they can compare their code. The SEI's proof-of-concept Source Code Analysis Laboratory (SCALe) combines multiple analyzers to compare clients' code to the C and Java secure coding standards. If the code conforms to the standards, the SEI issues the developer a formal certificate.

The SEI has been active in the evolution of existing language standards. An SEI secure coding expert chaired the PL22.11 C Programming Language Task Group of the International Committee for Information Technology Standards (INCITS). In 2011, ISO/IEC 9899:2011 [ISO 2011], the first major revision of the C language specification since 1999, incorporated several proposals from the Secure Coding Initiative to improve the security of the C language. The SEI experts have also been active in C++ standardization as well as other working groups and steering committees within the joint technical committee set up between the International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) to develop worldwide Information and Communication Technology (ICT) standards.

Addison-Wesley has published the secure coding standards for C and Java as books [Seacord 2008, Long 2011], as well as Java coding guidelines [Long 2013].The Secure Coding Initiative continues to evolve the standards for these languages. The SEI work has taken on new significance with Section 925 of the National Defense Authorization Act for Fiscal Year 2013, which requires evidence that the coding practices of government software development and maintenance organizations and contractors conform to secure coding standards approved by the DoD. Section 925 applies to coding practices during software development, upgrade, and maintenance.

The SEI Secure Coding Initiative is now leading a community effort to develop the secure coding standards for other common programming languages. Draft versions of C++ and Perl language standards are available, and secure coding standards for other languages, including Ada, SPARK, and C#, are in the stages of development.

The SEI's ultimate goal is to help developers make their software less vulnerable before it is released. To this

**The View from Others**

*In the Java world, security is not viewed as an add-on a feature. It is a pervasive way of thinking. …. A set of standard practices has evolved over the years. The Secure® Coding® Standard for Java™ is a compendium of these practices. These are not theoretical research papers or product marketing blurbs. This is all serious, mission-critical, battle-tested, enterprise-scale stuff.*

– James A. Gosling, Father of the Java Programming Language [Long 2013]

*CERT C/C++ Secure Coding Standard as the internal secure coding standard for all C/C++ developers. It is a core component of our secure development lifecycle. The coding standard described in this book breaks down complex software security topics into easy to follow rules with excellent real-world examples. It is an essential reference for any developer who wishes to write secure and resilient software in C and C++.*

– Edward D. Paradise, VP Engineering, Threat Response, Intelligence, and Development, CISCO. [Seacord 2008]

end, the SEI has developed training courses on secure coding and is working with the software development community to eliminate barriers to the broader adoption of the secure coding standards.

### 5.4.3 The Consequence: More Secure Products

One of the Secure Coding Initiative's farthest-reaching effort was its contribution to the ISO/IEC C-language specification. C language compilers, which C language developers use to compile their code, are based on the ISO/IEC C-Standard; so the initiative's security contributions to the specification will propagate to countless software products. The U.S. military, other government agencies, and system developers from industry have adopted CERT secure coding standards, and Siemens and Computer Associates have licensed the SEI's training courses on secure coding in C and C++. Many others have taken the SEI courses, including the Department of Homeland Security, Department of Energy, National Security Agency, Central Intelligence Agency, U.S. Navy, Cisco, and the Center for Financial Technologies in Russia.

**The View from Others**

*A must-read for all Java developers....Every developer has a responsibility to author code that is free of significant security vulnerabilities. This book provides realistic guidance to help Java developers implement desired functionality with security, reliability, and maintainability goals in mind.*

– Mary Ann Davidson, Chief Security Officer, Oracle Corporation [Long 2013]

Conformance testing helps influence the development community from the top down by giving software vendors the opportunity to turn technical success into business success. The SEI-issued "Conformance Tested" seal, indicating a developer's standard-conformant software, can be used by a vendor to promote the product, increasing the market value of provably more secure code.

Courses based on the *Secure Coding in C and C++* book [Seacord 2013] are taught at major universities and colleges; for example, Carnegie Mellon, Purdue, Stanford, Stevens Institute, University of Florida, University of Illinois, University of Pittsburgh, and University of Texas. Finally, TSP-Secure, an extension of TSP, requires the selection of coding standards during the requirements phase; TSP teams use application conformance testing as part of their own development process.

### 5.4.4 The SEI Contribution

Other organizations have also done work in secure software. For example, the MITRE Corp. has developed the Common Weakness Enumeration (CWE), a compilation of common software weaknesses, mapped to the CERT secure coding standards. The latest C Standard includes changes proposed by the Secure Coding Initiative to improve C-language security, which should be implemented by the more than 200 C compilers that conform to the C Standard. The SEI Secure Coding Initiative's work has also influenced a variety of code analyzer vendors, including LDRA (see "View from Others," above).

While the SEI's work in the C programming language has had the most public impact on software development, the secure coding standard for Java has also made significant contributions. Im-

provements in how developers use both languages have propagated to countless software products, including many in the DoD supply chain. The SEI secure coding experts also participated in the C Secure Coding Rules Study Group, whose work resulted in the publication of *ISO/IEC TS 17961(E), Information Technology—Programming Languages, Their Environments and System Software Interfaces—C Secure Coding Rules.* The Secure Coding Initiative's engagement with such international standards bodies improves the initiative's standards, processes, and influence.

### 5.4.5 References


[Businesswire 2008] "LDRA Ships New TBsecure™ Complete with CERT C Secure Coding Programming Checker." Businesswire, October 27, 2008. http://www.businesswire.com/news/home/20081027005019/en/LDRA-Ships-TBsecure-TM-Complete-CERT-Secure

[Dobbs 2009] "Secure Coding in C and C++." *Dr. Dobb's Journal.* September 3, 2009. http://drdobbs.com/cpp/219501214

[ISO 2011] International Standards Organization & International Electrotechnical Commission. "Programming Languages—C," International Standard 9899:2011. http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1570.pdf (2011).

[Keizer 2011] Keizer, Gregg. "Hackers Launch Millions of Java Exploits." *Computerworld*, November 19, 2011. http://www.infoworld.com/article/2621397/security/microsoft--hackers-launch-millions-of-java-exploits.html

[Long 2011] Long, Fred; Mohindra, Dhruv; Seacord, Robert C.; Sutherland, Dean F.; & Svoboda, David. *The CERT Oracle Secure Coding Standard for Java.* Addison-Wesley, 2011 (ISBN 0321803957).

[Long 2013] Long, Fred; Mohindra, Dhruv; Seacord, Robert C.; Sutherland, Dean F.; & Svoboda, David. *Java Coding Guidelines: 75 Recommendations for Reliable and Secure Programs.* Addison Wesley Professional, 2013 (ISBN 0-321-93315-X).

[Seacord 2008] Seacord, Robert C. *The CERT C Secure Coding Standard*. Addison Wesley Professional, 2008 (ISBN 0321563212).

[Seacord 2013] Seacord, Robert C. *Secure Coding in C and C++, 2nd Edition.* Addison Wesley Professional, 2013 (ISBN 0321822137).

## 5.5 Network Situational Awareness

### 5.5.1 The Challenge: Visibility of Large Networks

Operators of networks of any size struggle to understand the activity on it. The challenge is to understand what data needs to be captured, how that data should be collected, how that data should be stored, and ultimately how that data can be effectively analyzed to produce valuable metrics and identify problems.

### 5.5.2 A Solution: Network Situational Awareness Tools and Techniques

SEI network situational awareness (NetSA) experts develop ways to assist operators and analysts in understanding activity on their networks by using meaningful pictures of their large volumes of data. In developing an operational view of network attacks and network baselines and uncovering anomalies, the SEI has developed standards for describing network traffic, designed sensors and analysis tools, and uses both existing and SEI-developed network flow and intrusion detection technology.

In the early 1990s, the CERT Coordination Center at the SEI developed Argus, one of the first software-based network flow analysis tools, to support incident response activity. Argus provided a practical means to summarize full packet capture for security purposes and ultimately was used in traffic engineering.[36] In 2000, the Automated Incident Reporting to CERT (AirCERT) initiative released data conversion, sharing, and analysis tools (Analysis Console for Incident Data—ACID); improved open-source sensors (snort); and supported the development of Internet Engineering Task Force (IETF) standards establishing a data format for exchanging information on computer security incidents among response teams around the world.[37] The Department of Defense adopted SEI technologies in DoD-CERT, and General Services Administration's FedCIRC piloted them in federal civilian agencies. The concept of normalizing event data from different sources matured into the security information and event manager market, and the notion of collecting data from different organizations is known today as the security intelligence market.

In 2001, the SEI revisited network flow analysis by building the System for Internet Level Knowledge (SiLK) tool suite for the DoD to conduct security analysis not driven by known-bad signatures. The tool suite is a scalable system that enabled network forensics for all internet traffic in the DoD. As the SEI became more active with trending, profiling, and capacity planning, it developed IETF standards for this data [IPFIX 2011].

SEI advances have moved beyond netflow analytics. SEI engineers created Yet Another Flowmeter (YAF) [Inacio 2010], which leverages additional data sources, including application-level information—Domain Name System, Secure Socket Layer certificates, and application banners stored in the IPFIX standard format. Further, the SEI supports key sponsors in higher level areas, such as systems engineering, architecture, and overall Computer Network Defense program management.

---

36 Argus development continues outside the SEI, at QoSient.

37 Intrusion Detection Message Exchange Format (IDMEF) and Incident Object Description and Exchange Format (IODEF). Cover Pages [CP 2001] has further details and useful links.

### 5.5.3 The Consequence: Improved Situational Awareness with SEI Tools

SEI network situational awareness tools are actively used in operations on very large networks. The tools help DoD and federal agencies characterize network threats, assess the impact of security events, and identify vulnerable network infrastructure. Because the tools provide ongoing information about the normal traffic on the network, they support network analysts in assessing the effectiveness of their defensive actions and help with traffic engineering and capacity planning. The SEI flow detection methodologies are a practical means of providing context for particularly severe or long-lasting incidents, such as those found at NASDAQ [Schwartz 2011], and the means of enabling analysts to get a broad profile of network behavior before they perform detailed analysis.

The Centaur program is the largest system in the DoD for global situational awareness of the NIPRNet available to Tier 1 Computer Network Defense analysts. It gives DoD network and intelligence analysts a comprehensive means to uncover and measure both strategic and tactical network-security threat activity. Success in the DoD led to voluntary deployments in the federal civilian agencies by the Department of Homeland Security's US-CERT through the Special Access Programs initiative and, later, Einstein. The SEI analysis approach and tool suite are also used by the National Security Agency's National Threat Operations Center, Service computer security incident response teams (CSIRTs), Defense Information Systems Agency Regional CSIRTs, and the Defense Advanced Research Projects Agency, among others.

**The View from Others**

*MS-ISAC has been utilizing the SILK tools provided by the NETSA group of CERT/CC in order to monitor state and local government networks in the United States for almost a year now. We are amazed by the efficiency and scalability of the tools, and very grateful to the CERT/CC staff for providing an incredible support to keep our infrastructure running. SILK tools have enabled us to identify hundreds of security incidents affecting state/local governments which would otherwise go undetected. Thank you very much.*

– Multi-State Information Sharing and Analysis Center Manager

Open source release of YAF, SiLK, and associated tools has led to widespread adoption. Several commercial entities have incorporated the technology into their own products, for example, nPulse and 21st Century Technology's Lynxeon. Far more companies use SEI technology to help protect their own networks and the networks of their clients. Users include telecommunication providers, government defense contractors, and many other high-tech companies. The Multi-State Information Sharing and Analysis Center (MS-ISAC) uses SEI technology in its network monitoring. The DoD and DHS have used SEI tools and approaches to help security analysts profile and monitor U.S. government networks and systems for unauthorized access.

### 5.5.4 The SEI Contribution

Both netflow and intrusion detection technologies were in use before the SEI became involved. The SEI tools improved these technologies to operate at a large scale and with enhanced usability. The SEI took large-scale flow collection from initial creation to being the leading reference implementation supporting open standards for describing network traffic. Adoption of SEI tools and techniques can be found in the DoD, government (federal, state, local), in thriving commercial network monitoring services, and in security appliances.

In 2007, the Comprehensive National Cyber Initiative made the Einstein program mandatory for all federal civilian agencies.[38] US-CERT at DHS uses Einstein to meet the mandatory and administrative requirements for DHS to help protect federal computer networks and the delivery of essential government services. Einstein's impact was also mentioned in a *Federal Computer Week* article [Miller 2007], which describes an attack launched on Department of Transportation computers from Department of Agriculture computers that were infected with a computer worm. The unusual network traffic was discovered at the Department of Transportation network gateway because of Einstein.

Finally, the SEI has built a strong community of network analysts through its annual FloCon workshop, which brings together analysts from academia, the government, and private industry. Attendance has grown steadily since the first workshop in 2003; FloCon 2014 had 188 participants.

### 5.5.5 References

[CP 2001] Cover Pages Technology Reports. "Intrusion Detection Message Exchange Format." January 15, 2001. http://xml.coverpages.org/idmef.html

[IPFIX 2011] IETF Network Working Group. "Specification of the IP Flow Information eXport (IPFIX) Protocol." November 29, 2011. http://tools.ietf.org/html/draft-ietf-ipfix-protocol-rfc5101bis-00. The latest information on the protocol can be found at http://tools.ietf.org/wg/ipfix/ (2011).

[Inacio 2010] Inacio, Chris & Trammell, Brian. "YAF: Yet Another Flowmeter," 107-118. *Proceedings of Large Installation System Administration Workshop (LISA)*. San Jose, CA, November 2010. USENIX, 2010. https://www.usenix.org/legacy/events/lisa10/tech/slides/inacio.pdf

[Miller 2007] Miller, Jason. "Einstein Keeps an Eye on Agency Networks." *Federal Computer Week.* May 21, 2007. http://fcw.com/Articles/2007/05/21/Einstein-keeps-an-eye-on-agency-networks.aspx

[Schwartz 2011] Schwartz, Matthew J. "Nasdaq Confirms Servers Breached." *Information Week.* http://www.informationweek.com/news/security/attacks/229201276 (2011).

38 Einstein's mandate originated in the Homeland Security Act and the Federal Information Security Management Act, both in 2002, and the presidential directive named Homeland Security Presidential Directive (HSPD), which was issued on December 17, 2003.

## 5.6 Insider Threat

### 5.6.1 The Challenge: Cyber Attacks by Insiders

Insiders pose a challenging cybersecurity threat. They are trusted employees, former employees, or even contractors with access to internal systems and sensitive information; because they have (or recently had) authorized, trusted access, it is hard to protect against their malicious actions [Trembly 2011]. These actions include IT sabotage, fraud, theft of confidential or proprietary information, espionage, and threats to U.S. critical infrastructure [Gupta 2008]. The actions of a single insider have resulted in a range of impacts, including loss of staff hours, loss of reputation and customer trust, and financial damage so extensive that businesses have been forced to lay off employees or cease operation. Damage from insider incidents can have far-reaching repercussions, creating serious risks to public safety and national security, such as disruption of a service in a critical infrastructure, disclosure of classified information, or industrial espionage. Addressing insider threat is a challenge, as technological solutions alone are ineffective.

**The View from Others**

*They have a great insider threat research team up there; they've been working on this for over 10 years.*

– Dr. Ron Ross, National Institute of Standards and Technology [SEI 2011]

*CERT is offering a fantastic Insider Threat Workshop that will be of extreme benefit to anybody in the computer security industry.*

– Lauren Gerber, in PC1news.com [Gerber 2009]

### 5.6.2 A Solution: Insider Threat Research and Solutions

The SEI has become a center of expertise on identifying and mitigating insider threat [Kaplan 2011]. The SEI first investigated the malicious actions of insiders in 2000, when the DoD sponsored research to identify characteristics of the environment surrounding insider incidents in the military services and defense agencies. The findings guided ongoing efforts to reduce the threat to critical information systems in the DoD and its contractor community. This work was the beginning of an ongoing partnership between the SEI and the DoD's Defense Personnel Security Research Center (PERSEREC). The following year, the U.S. Secret Service National Threat Assessment Center (NTAC) and the SEI worked together to conduct a unique study of insider incidents—psychologists from NTAC and technical experts from the SEI examined insider cases both from a behavioral and a technical perspective. It was the first study that used this dual approach. In 2002, SEI security experts assisted the U.S. Secret Service (USSS) with the cyber aspect of its protection mission at the Salt Lake City Olympic Games, categorized as a National Security Special Event. In talking with the Olympic Committee and considering potential problems, SEI staff realized the criticality and extent of insider threat. Work with the Secret Service continued, and research was stepped up. In 2013, the Secret Service honored SEI insider threat experts for their "efforts and superior contributions" to USSS law enforcement responsibilities.[39]

In 2003 and 2004, the Department of Homeland Security (DHS) added its sponsorship to the insider threat study and to building a database of the valuable information collected during the

39 See http://www.sei.cmu.edu/newsitems/USSS-award.cfm

study. In 2011, the DoD began funding development of a version of the database for use by government researchers and law enforcement.[40] The studies [CSO 2004][41] and database development work led to the development of best practices [Cappelli 2012][42] and models [Moore 2008],[43] both funded by CyLab at Carnegie Mellon University. The models, which include behavioral traits and technical actions, reveal indicators that might alert an organization to the potential for malicious acts by insiders [Greene 2010]. Model development is ongoing, with models now available for insider IT sabotage, insider theft of intellectual property, and national security espionage. In 2011-12, development began on a fraud model with the USSS, Department of Treasury, and the financial sector (sponsored by DHS). In 2012, the SEI expanded its research and case collection process to include insider incidents that occur outside the United States as well as those perpetrated by insiders but without malicious intent. This expansion has allowed international [Flynn 2013] and unintentional insider threat studies [CERT IT Team 2013], providing a more complete picture of the threat posed by insiders to organization's critical assets.

The SEI conducts workshops on how to apply the practices and models. To extend its impact, the SEI published *The CERT Guide to Insider Threats: How to Prevent, Detect, and Respond to Information Technology Crimes,* which combines 10 years of research into one practical guide [Cappelli 2012]. SEI insider threat experts also worked with the Carnegie Mellon Entertainment Technology Center to create a prototype interactive virtual simulation tool—essentially a video game—teaching insider threat mitigation.

**The View from Others**

*New research from Carnegie Mellon University's Software Engineering Institute provides further evidence why information security isn't just the problem of an enterprise's IT and IT security organization but of its top non-IT leadership as well.*

– Eric Chabrow, in The Public Eye, a government security blog [Chabrow 2011]

The Insider Threat Study is an excellent example of collaboration between the federal government and private sector to safeguard the financial payment systems of the United States.

– Ryan Moore, Assistant to the Special Agent in Charge, USSS, in a press release on an award to SEI Insider Threat staff members [SEI 2013]

In response to community need, the insider threat team redirected its research toward solution-oriented activities. With seed funding from Carnegie Mellon's CyLab, the team developed insider threat assessments (ITAs), which are based on more than 4,000 indicators (organized into more than 130 categories) identified in the Insider Threat Database. DHS funded development of the ITA, Version 2,

40 In this version of the database, identities were made anonymous.

41 The first study [CSO 2004], and subsequent studies and analyses are available on the CERT website (http://www.cert.org/insider_threat).

42 Practices can be found in a set of reports available at http://www.cert.org/insider-threat/publications/index.cfm. One of the best known is the "*Common Sense Guide*," now in its fourth edition [Silowash 2012].

43 The report describes the first model developed [Moore 2008]. This model and others can be found at http://www.cert.org/insider-threat/research/Modeling-and-Simulation.cfm.

which SEI insider threat experts use to collect data that organizations can use as a benchmark. As a supplement to the face-to-face workshops already in place, insider threat experts developed exercises for STEPfwd, an SEI web-based platform that enables participants in multiple locations to work together on simulations of the latest threats. Both government and private industry have taken advantage of the ITAs and the online exercises.

In 2009, the SEI set up the CERT Insider Threat Lab, where its technologists could test existing technical solutions for the insider threat problem and identify new or refined solutions in gap areas. For example, research is underway on methods that help cloud service providers deal with insider threats [Porter 2013]. The lab is developing new technical controls for government and industry and making the controls available online.[44]

### 5.6.3 The Consequence: Improved Insider Threat Detection and Response

Using SEI results, information assurance staff and counterintelligence analysts have implemented technical controls for catching insiders. The DoD and federal civilian agencies can now identify insider threat proactively using SEI techniques and tools instead turning to forensics after the crime. In case of an attack, organizations are armed with policies, practices, and technical measures to help with recovering from the attack, identifying the perpetrator, and implementing new measures for improved incident management in the future.

### 5.6.4 The SEI Contribution

Law enforcement had long been profiling miscreants in general but did not focus on insider threat to computer systems and the information that resides on them. Similarly, network security experts focused on protecting against the technical attacks from outside the perimeter rather than attacks from inside. The SEI contribution was to examine cyber attacks by insiders from both technical and behavioral perspectives and to use real-life cases in this research. The ultimate goal of SEI insider threat research is to help all organizations, including the DoD, federal agencies, and critical sector industries, prevent insider attacks and, if there is such an attack, to provide these organizations with the tools, techniques, and methods that enable them to detect and respond to the illicit activity early, thus minimizing the impact to critical assets. This goal is well on its way to realization.

### 5.6.5 References

[Cappelli 2012] Cappelli, Dawn; Moore, Andrew P.; & Trzeciak, Randy. *The CERT Guide to Insider Threat: How to Prevent, Detect, and Respond to Information Technology Crime.* Addison Wesley Professional, February 2012 (ISBN 0321812573).

[CERT IT Team 2013] *Unintentional Insider Threats: A Foundational Study* (CMU/SEI-2013-TN-022). Software Engineering Institute, Carnegie Mellon University, 2013. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=58744

[Chabrow 2011] Chabrow, Eric. “Obama Reiterates Cybersec as a Priority.” *The Public Eye.* December 1, 2011. http://www.inforisktoday.com/blogs/public-eye-b-13/p-21

44 See http://www.cert.org/insider-threat/research/controls-and-indicators.cfm

[CSO 2004] CSO Magazine, in cooperation with the U.S. Secret Service and CERT Coordination Center. *2004 E-Crime Watch Survey: Summary of Findings,* 2004. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=53389

[Flynn 2013] Flynn, Lori; Huth, Carly; Trzeciak, Randall; & Buttles-Valdez, Palma. *Best Practices Against Insider Threats in All Nations* (CMU/SEI-2013-TN-023). Software Engineering Institute, Carnegie Mellon University, 2013. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=59082

[Gerber 2009] Gerber, Lauren." Insider Threat Workshop To Be Held By Cert This May!" *PC1news.com.* March 24, 2009. http://www.pc1news.com/news/556/cert-comes-to-the-rescue.html

[Greene 2010] Greene, Tim. "Fight Insider Threat with Tools You Already Have," *Network World* (September 27, 2010). http://www.networkworld.com/Home/tgreene.html

[Gupta 2008] Gupta, Upasana. "Tackling the Insider Threat," *Bank Info Security*. November 6, 2008. http://www.bankinfosecurity.com/articles.php?art_id=1042

[Kaplan 2011] Dan Kaplan. "Internal Review," *SC Magazine 22*, 2 (February 2011): 22-26. http://media.scmagazine.com/documents/91/scc_0211v2_22553.pdf

[Moore 2008] Moore, Andrew; Cappelli, Dawn; & Trzeciak, Randall. *The "Big Picture" of Insider IT Sabotage Across U.S. Critical Infrastructures* (CMU/SEI-2008-TR-009). Software Engineering Institute, Carnegie Mellon University, 2008. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=8703

[Porter 2013] Porter, Greg. *Cloud Service Provider Methods for Managing Insider Threats: Analysis Phase I* (CMU/SEI-2013-TN-020). Software Engineering Institute, Carnegie Mellon University, 2013. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=69709

[SEI 2011] Software Engineering Institute, Carnegie Mellon University. *Weekly Headlines*. 2011. https://mahara.org/artefact/file/download.php?file=177333&view=51792

[SEI 2013] Software Engineering Institute, Carnegie Mellon University. *United States Secret Service Honors CERT Insider Threat Team Members.* 2013. http://www.sei.cmu.edu/news/article.cfm?assetid=53981&article=114&year=2013

[Silowash 2012] Silowash, George; Cappelli, Dawn; Moore, Andrew; Trzeciak, Randall; Shimeall, Timothy; & Flynn, Lori. *Common Sense Guide to Mitigating Insider Threats, Fourth Edition* (CMU/SEI-2012-TR-012). Software Engineering Institute, Carnegie Mellon University, 2012. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=34017

[Trembly 2011] Trembly, Ara C. "We Have Seen the Enemy, and He Works for Us." *Information Management Online.* February 16, 2011. http://www.information-management.com/news/data_security_information_management_GRC-10019745-1.html (2011).

## 5.7 Information Security Assessments

### 5.7.1 The Challenge: Managing Risks to Enterprise-Wide Information Security

Before the era of pervasive computing, the major enterprise assets were tangible, such as buildings, equipment, and physical products. Now intangibles are often the most critical assets [Webber 2000]—intangibles such as intellectual property, patient records, customer data, and other personally identifiable information. When a security breach compromises critical assets, an organization can suffer not only monetary loss but also loss of proprietary information, reputation, and the public's trust. Many government and commercial organizations have not identified or placed a value on their intangible assets or assessed the risk to those assets, so they cannot know if their important information is adequately protected or if resources are used to protect relatively unimportant information. The lack of effective risk identification and management has an impact on both the organization and on U.S. economic security.

### 5.7.2 A Solution: Managing Risks to Enterprise-Wide Information Security

The SEI began helping organizations identify software development risks in the early 1990s through its Software Risk Evaluation (SRE). Prompted by the desire to help organizations better identify cybersecurity risks, the SEI subsequently developed the *Information Security Evaluation* (ISE). Drawing from SRE experiences, developers combined interviews with management and staff (separately) with a technology evaluation to help organizations identify their assets and determine their information security risks. The ISE team provided practical guidance along with its findings.

The SEI subsequently documented best practices in CERT security improvement modules—modular documents that contain concrete guidance for analyzing and improving specific aspects of security on networked systems. The modules were developed from 1996 to 2001 and were subsequently published in a book [Allen 2001].[45] In parallel, starting in 1997, the SEI created a new approach for managing cybersecurity risk—the Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE). Development was prompted by a combination of SEI experience with the ISE and the Defense Health Information Assurance Program (DHIAP).[46] The SEI goal was a self-directed risk assessment as part of the DoD effort to comply with the data security requirements defined by the Health Insurance Portability and Accountability Act (HIPAA) of 1996. The OCTAVE method incorporated the principles of ISE and continuous risk management. The method was defined in a 1999 OCTAVE framework, the blueprint for the full method, released in 2001. First piloted with an SEI evaluation team, the OCTAVE method [Alberts 2003] became a self-directed risk evaluation that meets the unique needs of each organization. It balances the organization's critical information assets, business needs, threats, and vulnerabilities, and also benchmarks the organization against known good practice.

45 The *CERT Guide to System and Network Security Practices* has been translated into four languages.

46 DHIAP was a small consortium that included the SEI and the Advanced Technology Institute (ATI) of the South Carolina Research Authority (SCRA) and was overseen by a group from the Telemedicine Advanced Technology Research Center (TATRC) from Fort Detrick, Maryland.

People who work in small organizations liked the OCTAVE approach but needed a streamlined method to accommodate their staff size, schedules, and budgets. In response, the SEI developed OCTAVE-S [Alberts 2005].[47] While remaining consistent with OCTAVE principles, OCTAVE-S provides organizations of fewer than 100 people with an efficient, inexpensive approach to identifying and managing information security risks. Both OCTAVE and OCTAVE-S are supported with guidance, worksheets, and questionnaires. The SEI developed another alternative, OCTAVE Allegro [Caralli 2007], in response to OCTAVE users who were looking for a more information-centric method that could be institutionalized at the operational unit level. Allegro helps businesses identify their information assets and determine how those assets are at risk by putting them in the context of "containers"—places where information is stored, transmitted, or processed. While Allegro can be used in a collaborative workshop style like the original OCTAVE methods, it is also well suited for individuals who want to perform a risk assessment without extensive organizational involvement or expertise. Allegro developers reduced the amount of risk analysis and IT knowledge needed, and simplified instructions and worksheets.

**The View from Others**

*Conducting a security risk analysis has long been a requirement of the HIPAA Security Rule and is also necessary to achieve Meaningful Use regarding the use of electronic health records ("EHR") systems. OCTAVE Allegro provides us with a useful framework for assessing risks to ePHI, including EHR's, while at the same time providing the evidentiary requirements necessary for regulatory compliance. Our clients need something that is easy to deploy, repeatable, underpinned by good practice and OCTAVE provides this.*

– Greg Porter, Founder, Principal Consultant, Allegheny Digital

### 5.7.3 The Consequence: Enterprise Risk Management and Security Improvement

Organizations using the OCTAVE product suite have control over their information security activities [Shantamurthy 2011]. Managers can develop a protection strategy that is appropriate for their particular organizations' mission and priorities, a strategy that addresses policy, management, administrative, and technological aspects, among others. As a result of SEI evaluations, government and commercial organizations have a clearer view of their information security risk and control over their security posture. They manage their risk through improvement efforts and periodic assessments, which they schedule and perform at their own discretion. Many organizations establish a multidisciplinary team that can perform the follow-up assessments and act as a focal point for the improvement efforts. Important organizational and individuals' information are protected. As a result, the organizations improve not only their own risk profile but also that of the sectors to which they belong—thus contributing to national security.

One example of information protection and security improvement is the use of the OCTAVE method and OCTAVE Allegro by various agencies of the county government of Clark County, Nevada. Clark County adopted the OCTAVE method as a way to comply with the federal HIPAA

47 The development of OCTAVE-S was sponsored by the SEI Technology Insertion, Demonstration, and Evaluation (TIDE) program, created to help small manufacturing enterprises adopt state-of-the-practice technologies.

requirements to protect the privacy of personal information collected through the county's social services. It later moved to OCTAVE Allegro, which SEI staff members developed in response to the county's need. Clark County went on to take SEI train-the-trainer classes and is implementing OCTAVE Allegro organization-wide at the operational unit level.

The U.S. Army and Air Force are using the OCTAVE method [Coleman 2003]. OCTAVE is used by the DHIAP, other federal programs, and the commercial sector. The Telemedicine and Advanced Technology Research Center (TATRC) taught the OCTAVE method for use at DoD medical treatment facilities around the globe.

### 5.7.4 The SEI Contribution

Various information technology assessments were used before the SEI developed enterprise-wide evaluations that stress information security as it relates to each organization's mission, critical assets, and priorities. Although other methods are available today [Violino 2010], OCTAVE's focus on continuous risk management has been described by IEEE as a de facto standard. The OCTAVE method is used for HIPAA compliance in both the public and private sectors.

The SEI has made a major contribution to the DoD with the development of the OCTAVE method, particularly in the medical area. The Surgeon General for the Army and Air Force recognized the OCTAVE method as the recommended best practice for HIPPA risk assessments.

### 5.7.5 References

[Alberts 2003] Alberts, Christopher & Dorofee, Audrey. *Managing Information Security Risks: The OCTAVE Approach*. Addison-Wesley Professional, 2003 (ISBN 0321118863).

[Alberts 2005] Alberts, Christopher; Dorofee, Audrey; Stevens, James; & Woody, Carol. *OCTAVE-S Implementation Guide, Version 1* (CMU/SEI-2004-HB-003). Software Engineering Institute, Carnegie Mellon University, 2005. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=6795

[Allen 2001] Allen, Julia H. *The CERT Guide to System and Network Security Practices.* Addison-Wesley Professional, 2001 (ISBN 020173723X).

[Caralli 2007] Caralli, Richard; Stevens, James; Young, Lisa; & Wilson, William. *Introducing OCTAVE Allegro: Improving the Information Security Risk Assessment Process* (CMU/SEI-2007-TR-012). Software Engineering Institute, Carnegie Mellon University, 2007. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=8419

[Coleman 2003] Coleman, Jonathan. "Execution of a Self-Directed Risk Assessment Methodology to Address HIPPA Data Security Requirements." Society of Photo-Optical Instrumentation Engineers (SPIE), 2003. http://spie.org/Publications/Proceedings/Paper/10.1117/12.480653

[Shantamurthy 2011] Shantamurthy, Dharshan. "OCTAVE Risk Assessment Examined Up Close." *TechTarget*. http://searchsecurity.techtarget.in/tip/OCTAVE-risk-assessment-method-examined-up-close (2011).

[Violino 2010] Violino, Bob. “IT Risk Assessment Frameworks: Real-World Experience.” *NetworkWorld*. May 3, 2010. http://www.networkworld.com/news/2010/050310-it-risk-assessment-frameworks-real-world.html

[Webber 2000] Webber, Alan M. “New Math for a New Economy.” *Fast Company 31* (January-February): 214. http://www.fastcompany.com/magazine/31/lev.html

## 5.8 Cybersecurity Engineering

### 5.8.1 The Challenge: Software Security Assurance

Software assurance is the confidence that software is free from vulnerabilities—either intentionally designed into the software or accidentally inserted at any time during its lifecycle—and that the software functions in the intended manner [CNSS 2010]. Developing and acquiring software with some level of assurance that it will perform as intended requires addressing a broad range of essential activities, such as requirements analysis, system architecture and design, program development and integration, testing, maintenance, and project management. Thus, software assurance requires building security into the system early in the lifecycle, with continued emphasis throughout the lifecycle. As DoD and all federal agencies rely on software-based systems, software vulnerabilities become an Achilles heel that can allow potential access for malicious or inadvertent reconnaissance, exploitation, subversion, sabotage, disclosure and denial of system services. It is essential to understand how to build, acquire, and operate dependable, trusted, survivable systems that provide a measured level of software assurance.

### 5.8.2 A Solution: Build In Security from the Start

In 1998, the SEI began looking into engineering solutions that result in more secure as-shipped products. SEI researchers focused on the earlier phases of the software development lifecycle, identified gap areas, and did exploratory work to find solutions. In addition to drawing on their own knowledge, SEI security experts assembled input from SEI customers and from participants in Information Survivability Workshops (ISWs), held annually from 1997 to 2001.[48] These ISWs identified a range of gap areas that motivated early research.

Early work was concentrated on analytical methods and supporting tools. This research included development of a simulation language, Easel [Stojkovic 2005]. Easel was designed to model network attacks and assess the effectiveness of mitigation techniques. It served as a demonstration of the technology, and the research resulted in a better understanding of the potential impacts of emergent algorithms on system security. Function Extraction [Pleszkoch 2004] addressed the need for a practical means of determining what sizable programs do in all circumstances of use; it focused on automatically extracting as-built functionality from the executable language. Its initial application was to extract the functionality of malicious code for analysts seeking to create effective countermeasures. The Function Extraction library of software tools was approved for public release in 2012. The Survivable Network Analysis (SNA) method [Mead 2000] was developed as a process for systematically assessing the survivability properties of both proposed and existing systems; the analysis could be done at the lifecycle, requirements, or architecture level. (SNA was later renamed Survivable Systems Analysis [SSA] to reflect the emphasis on analyzing systems rather than networks.) SNA was the precursor of Software Quality Requirements Engineering (SQUARE) [Mead 2005], a nine-step process for building security in at the requirements phase.

48 Co-sponsored by the SEI and IEEE, each ISW brought together researchers and practitioners to discuss survivability, related problems, and promising approaches to solutions. Survivability is the ability of a system to limit damage and continue critical functions even when attacked. The dedicated ISWs were discontinued when other conferences began to including include survivability tracks.

Since its initial development in 2005, SQUARE has been adapted to acquisition and extended to include privacy; tools have been developed to support its use.

In addition to developing analysis methods and tools, SEI researchers explored ways to identify organizational and systemic risks that could affect software security. The Mission Assurance Analysis Protocol [Alberts 2005] was initiated to identify and understand inherent operational risks when management control of work processes is distributed among multiple organizations. The Vendor Risk Assessment and Threat Evaluation [Lipson 2001] was developed to assess vendor capabilities as an indicator of product quality. In late 2006, the SEI began research on supply chain integrity [Ellison 2010a]; SEI experts defined risk management approaches (see paper by Croll [Croll 2013]) that can be used during acquisition, development, and transit, as well as when components are integrated with other software and when changes occur in the environment and in attack techniques after deployment. To address complexity, the SEI developed two frameworks, starting in 2008—the Survivability Analysis Framework [Ellison 2010b], a structured view of people, activities, and technology that helps organizations characterize the complexity of dynamic multi-system and multi-organizational business processes and the Software Assurance Modeling Framework [Siviy 2009], which enables organizations to tie their current environment to operational needs and identify areas where they can improve assurance. In 2010, the SEI began work on a risk-based approach for measuring and monitoring the security characteristics of interactively complex, software-reliant systems across the lifecycle and supply chain. The Integrated Measurement Analysis Framework [Alberts 2010] integrates performance data for individual components to provide a consolidated view of system performance. The Mission Risk Diagnostic [Alberts 2012] analyzes the risk to the system as a whole for a comprehensive view of the overall risk to a system's mission.

**The View from Others**

*Our company provided them with an opportunity to assess a many-faceted product and they responded graciously by sharing the different techniques they used to analyze the security aspects of our application. Their results gave us insight that has since influenced our application development and configuration.*

– SQUARE client, a software development company

*We identify 23 activities that are essential to engineer complete and detailed security requirements. We use these 23 activities as a basis to compare five different requirements engineering processes. Our analysis shows that SQUARE incorporates more of these activities than other processes.*

– Muhammad Umair Ahmed Khan and Mohammed Zulkernine [Khan 2009]

In 2013, with DHS sponsorship, the SEI created a cybersecurity risk management strategy to aid alert originators planning to use the new wireless emergency alerting (WEA) capability implemented by the Federal Emergency Management Agency (FEMA) in April 2013 [Woody 2013]. They can use the strategy throughout WEA adoption, operations, and sustainment to decrease vulnerability to attack and manage risk in the face of changing threats. As part of this effort, the SEI is also working with the developers of alert originator software to increase WEA cybersecurity.

Recognizing that the principle of building security in at the start had to extend to the workforce, DHS sponsored the SEI to build a model curriculum for software assurance education. The SEI

also developed course descriptions and other resources. In 2011, the master's curriculum was recognized by the IEEE and the Association for Computing Machinery as the model curriculum for a master's degree program in software assurance.

### 5.8.3 The Consequence: Improved Software Development and Acquisition Practices

SEI tools, techniques, methods, and analysis are raising the level of awareness of software developers, acquirers, and system managers [Allen 2008]. Security and privacy can now be clearly defined in software requirements, helping to ensure these qualities are incorporated from the start. The use of SEI frameworks helps organizations to increase their confidence that operational mission and critical work processes can be successfully executed in the presence of stress and possible failure, and helps them to identify areas where they can apply policy, practices, and technology options to improve assurance. The risks inherent in supply chains can be assessed, reduced, and mitigated. Risk-based measurement techniques increase organizations' understanding of their software assurance situation and enable them to make effective improvements. The cybersecurity risk management strategy enables emergency alert originators to mitigate risks so that alerts are sent with proper authorization, accurately, and on time, every time.

The ultimate consequence is improved national security, with increased assurance that software will operate as expected for essential government services and the nation's critical infrastructure and with reduced risk and impact of successful cyber attacks.

### 5.8.4 The SEI Contribution

In seeking ways to prevent vulnerabilities rather than simply react to them, the SEI leveraged the software community's identification of gap areas in software assurance research and the knowledge gained in its CERT Coordination Center's reactive work on security breaches and software vulnerabilities. Some projects are unique approaches to software assurance; others adapt technology and techniques from other software-related areas. Along with the SEI research in this area, the software industry has recognized that security must be incorporated into product and systems development: for example, Microsoft's Security Development Lifecycle and Cigital's Build Security in Maturity Model. Likewise, the national Institute of Standards and Technology (NIST) and the Object Management Group (OMG) are developing standards and guidelines for addressing security in software development.

The SEI works with DHS, DoD agencies and organizations, and defense contractors to raise awareness of software assurance opportunities and requirements and to help them take action to build security into products early in the software development lifecycle. The SEI addresses the nation's need for increased software assurance expertise by offering training in SEI techniques and a curriculum[49] to prepare future software assurance experts. The institute reaches out to the community of software developers and acquirers by managing and contributing content to DHS websites—*Build Security In* (BSI) and the *Software Assurance (SwA) Community Resources and Information Clearinghouse* (CRIC). SEI experts also work with the software assurance community through DHS Software Assurance Working Groups.

49 See http://www.cert.org/mswa

### 5.8.5 References

[Alberts 2005] Alberts, Christopher & Dorofee, Audrey. *Mission Assurance Analysis Protocol (MAAP): Assessing Risk in Complex Environments* (CMU/SEI-2005-TN-032). Software Engineering Institute, Carnegie Mellon University, 2005. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=7505

[Alberts 2010] Alberts, Christopher; Allen, Julia; & Stoddard, Robert. *Integrated Measurement and Analysis Framework for Software Security* (CMU/SEI-2010-TN-025). Software Engineering Institute, Carnegie Mellon University, 2010. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9369

[Alberts 2012] Alberts, Christopher & Dorofee, Audrey. *Mission Risk Diagnostic (MRD) Method Description* (CMU/SEI-2012-TN-005). Software Engineering Institute, Carnegie Mellon University, 2012. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=10075

[Allen 2008] Allen, J.; Barnum, S.; Ellison, R.; McGraw, G.; & Mead, N. *Software Security Engineering: A Guide for Project Managers.* Addison-Wesley, 2008 (ISBN-13:978-0-321-50917-8).

[CNSS 2010] Committee on National Security Systems. *National Information Assurance Glossary.* CNSS Instruction No. 4009, April 26, 2010. http://www.ncix.gov/publications/policy/docs/CNSSI_4009.pdf

[Croll 2013] Croll, Paul. "Managing Supply Chain Risk – Understanding Vulnerabilities in the Code You Buy, Build, or Integrate." *IEEE Software Technology Conference (STC 2013).* Salt Lake City, Utah, April 8-11, 2013. IEEE, 2013.

[Ellison 2010a] Ellison, Robert & Woody, Carol. "Considering Software Supply-Chain Risks." *CrossTalk 23,* 5 (September/October 2010): 9-12.

[Ellison 2010b] Ellison, Robert & Woody, Carol. *Survivability Analysis Framework* (CMU/SEI-2010-TN-013). Software Engineering Institute, Carnegie Mellon University, 2010. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9323

[Khan 2009] Kahn, M. U. A. & Zulkernine, M. "On Selecting Appropriate Development Processes and Requirements Engineering Methods for Secure Software" 353-358. *33rd Annual IEEE International* Computer Soft*ware and Applications Conference, 2009 (COMPSAC '09*) *(Volume: 2).* Seattle, WA, July 2009. IEEE, 2009.

[Lipson 2001] Lipson, Howard; Mead, Nancy; & Moore, Andrew. *Can We Ever Build Survivable Systems from COTS Components?* (CMU/SEI-2001-TN-030). Software Engineering Institute, Carnegie Mellon University, 2001. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=5581

[Mead 2000] Mead, Nancy; Ellison, Robert; Linger, Richard; Longstaff, Thomas; & McHugh, John. *Survivable Network Analysis Method* (CMU/SEI-2000-TR-013). Software Engineering Institute, Carnegie Mellon University, 2000. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=5241

[Mead 2005] Mead, Nancy; Hough, Eric; & Stehney II, Ted. *Security Quality Requirements Engineering* (CMU/SEI-2005-TR-009). Software Engineering Institute, Carnegie Mellon University, 2005. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=7657

[Pleszkoch 2004] Pleszkoch, Mark G. & Linger, Richard C. "Improving Network System Security with Function Extraction Technology for Automated Calculation of Program Behavior," 90299. *Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04)*. Waikoloa, HI, January 5-8, 2004. IEEE, 2004.

[Siviy 2009] Siviy, J.; Moore, A.; Alberts, C.; Woody, C.; & Allen, J. "Value Mapping and Modeling SoS Assurance Technologies and Assurance Supply Chain," 236-240. *Proceedings of the IEEE International Systems Conference*. Vancouver, Canada, March 23-26, 2009. IEEE, 2009.

[Stojkovic 2005] Stojkovic, V.; Steele, G.; & Hyousseu, C. "Implementations of Some Classical Fundamental Algorithms Based on Actor-Oriented Data Structures and Actors in the Easel Programming Language," 438-443. *Proceedings of the International Conference on Integration of Knowledge Intensive Multi-Agent* Systems. Waltham, MA, April 18-21, 2005. IEEE, 2005.

[Woody 2013] Woody, Carol. "Mission Thread Security Analysis: A Tool for Systems Engineers to Characterize Operational Security Behavior." *INCOSE Insight 16*, 2 (July 2013): 37-40.

# 6 Software Engineering Methods

# 6.0 Introduction to Software Engineering Methods

## 6.0.1 Demands of Increasing Reliance on Software Systems

The SEI strives to anticipate and respond to the implicit and explicit demands of government-acquired systems as the increasing reliance on software evolves. Software engineering methods and practices have played a critical role in the SEI's response to these changing needs, ranging from the development of new understanding on the part of our customers to the development of new methods and tools that enable customers to improve their practices.

## 6.0.2 Evolving Software Configuration Management

One of the major expectations when the SEI began was that software engineering would be supported by a collection of effective methods and tools. Indeed, an integrated software development environment was a significant component of the Ada program [Buxton 1981] and, later, in the Software Technology for Adaptable Reliable Systems (STARS) Program [Druffel 1983]. In the mid-1980s, there were several commercially developed Ada development environments in the early stages of availability. The SEI initiated an Ada Environment Evaluation effort. In forming the plans for evaluating these environments, the SEI saw that each environment approached software configuration management differently. After some initial research, the SEI concluded that although configuration management was an essential component of any software development environment, the field lacked a solid theoretic basis. Therefore, the SEI launched an initiative in software configuration management aimed at rectifying this gap. Following a series of SEI-led workshops and studies, software configuration management concepts and key practices were outlined and given a form on which the software community could base the development of tools, methods, and practices.

## 6.0.3 Developing Community Standards: Computer-Aided Software Engineering

Ada and its development environments were on the vanguard of a growing government interest in the value of community standards. The Navy had a long history in standards and supporting tool sets. In the late 1980s, the Navy initiated the Next General Computer Resources (NGCR) program, a substantial effort to establish community standards for various aspects of Navy systems, including buses, networks, operating systems, and software engineering environments. The SEI was positioned to assist and collaborated with the NGCR program, establishing a Computer Aided Software Engineering (CASE) initiative in 1990. The SEI embarked on the development and codification of a body of expertise on CASE environment usage and adoption practices, including integration and reference models. This work was based on a coordinated set of studies and pragmatic experiments, reinforced by direct customer engagements, such as with the NGCR program [Brown 1993]. The CASE effort was instrumental in paving the way for the routine use of development and integration environments in government programs.

## 6.0.4 Developing Community Standards: Open Systems Engineering

Underlying the NGCR program's intent to form community standards for Navy systems were the concepts of open systems and open system architectures. Though there was growing use of open systems within the commercial sector, there was limited understanding of how open system concepts would apply within government programs and government acquisition constraints. The Navy rightly recognized the need for two key elements: commercial standards that accommodated

government needs, and guidance for transitioning open systems concepts into practice. The Navy approached the SEI to assist with both elements. The SEI participated on the IEEE POSIX standards working groups, contributing to the development of open systems standards such that government requirements were accommodated. In addition, the SEI developed a comprehensive course—the first of its kind—to educate government professionals on the concepts and practices of open systems engineering. Course development started in 1993, with an initial delivery in 1994. When the DoD Open Systems Joint Task Force (OS-JTF) was formed in 1995, one of its first activities was to attend the SEI course. Since then, the SEI has delivered the course to all three U.S. DoD services; several U.S. federal agencies, such as the GAO, and defense organizations in Canada, the United Kingdom, Australia, and New Zealand. Today, open systems and open system concepts are still used in defense systems, and the SEI executive-level course is still delivered in response to customer demand.

### 6.0.5 Aiding Understanding of Expanding Technology

As the field of software engineering matured in the 1990s, SEI customers increasingly needed to comprehend and apply the ever-expanding array of software engineering concepts, methods, tools, and techniques. The SEI responded to a request from the Air Force acquisition community to develop the Software Technology Reference Guide (STRG), which provided the Air Force with an accessible and reliable source for basic information on a wide variety of technologies.

A further trend in the early 1990s was an emerging interest in reengineering within the SEI customer base. Faced with declining budgets and a significant investment in existing software systems, government programs recognized the need to leverage legacy software, rather than to develop new software from scratch, as they tried to meet changing mission needs and accommodate newer technologies. To compound the problem, these legacy systems were often inadequately documented and poorly structured. Further, there was an unrealistic expectation that emerging technology would allow for a straightforward, automated approach to reengineering these legacy systems. The SEI started a reengineering effort to assist its customers in understanding the issues and realities of re-engineering legacy assets and migrating these into new systems. In collaborating with government programs, the SEI developed methods, such as Options Analysis for Reengineering (OAR), which assists programs in more effectively identifying and mining legacy software components. Because of the strength and quality of OAR, it has had further impact as the basis for the key SEI approach for service-oriented systems, the SOA Migration, Adoption, and Reuse Technique (SMART), discussed in Building and Fielding Interoperating Systems.

### 6.0.6 Managing and Engineering COTS-Based Systems

In 1994, a Defense Science Board (DSB) study on the use of commercial products (also referred to as commercial off-the-shelf, or COTS, products) within defense systems made strong recommendations for the use of COTS products [DSB 1994]. The study provided the impetus for the SEI to create the COTS-Based Systems (CBS) initiative, which focused on understanding the specific needs within the defense community. The initiative built on the expertise from the CASE environments work, which was exploring the use of COTS products in software engineering environments. Over the next few years, the SEI led and actively contributed to an international community that developed key concepts, methods, and practices for managing and engineering systems that were built using a variety of COTS products. The SEI developed an integrated set of training courses to increase the government's awareness, understanding, and skills. These courses

were delivered to Army and Air Force programs, along with civil and defense agencies. One program invited the SEI to evaluate its CBS practices to highlight possible deficiencies. As a result, the SEI saw the need for a more robust example of a development process. This provided the genesis to create the Evolutionary Process for Integrating CBS (EPIC) that subsequently was licensed by IBM Rational and used as the basis for a CBS plug-in for their Rational Method Composer 7.5 process tool [IBM 2010].

### 6.0.7 Assurance Cases: Addressing Systems of Systems Challenges

By the beginning of the new millennium, another evolution in approaches to implementing software systems was taking place in defense and federal government communities: net-centric warfare, interoperability, and the creation of systems of systems (SoS). The SEI created the Integration of Software-Intensive Systems initiative. This initiative leveraged the rich history with COTS-based systems, open systems, reengineering, and software architecture principles to tackle the growing need for reliable and timely interoperation across multiple systems and organizations. Since 2004, the SEI has worked with an international community of collaborators to create practical concepts, frameworks, and methods that enable SEI customers to effectively evolve to the Global Information Grid (GIG) and potentially realize the benefits of net centricity. As service-orientation is one approach for net-centric and SoS implementations, the SEI developed extensive expertise in service-oriented architecture (SOA), creating courses for government personnel, developing a family of products (including SMART) to support the migration from traditional to SOA-based systems, and leading the international community in the development of a SOA research agenda.

Net centricity and systems of systems bring new engineering challenges to government communities on a scale not previously seen. Systems of systems are not built from scratch with a single organizational entity in control. Rather, they evolve from (parts or all of) existing systems, in varying stages of development and fielding, that are engineered, managed, and funded across multiple organizations, usually with no single governing entity. The SEI has developed methods and techniques to assist programs to gain insights into critical perspectives and into expectations about user demands that exceed those typical in product-centered engineering.

A concern for DoD systems was the need to shorten the certification process for safety, system reliability, or security. Traditional software and systems engineering techniques, including conventional test and evaluation approaches, were unable to provide the justified confidence needed. Consequently, a methodology to augment testing and evaluation was required. The SEI's experience in areas related to DoD certification needs through its work on rate monotonic analysis and Simplex led to a more general interest in performance-critical systems. Concurrently, the SEI was pursuing software issues associated with fault-tolerant computing and systems of systems. Because of their size, complexity, and continuing evolution, and because net-centric systems can exhibit undesired and unanticipated emergent behavior, the SEI decided on an approach using assurance cases.

An assurance case provides a means to structure the reasoning that engineers use implicitly to gain confidence that systems will work as expected. The SEI's early work on assurance cases was funded by NASA and, although NASA has not yet embraced the idea, NASA research continues on assurance case approaches. As a result of work with the SEI [Weinstock 2009], the U.S. Food

and Drug Administration issued draft guidance to manufacturers recommending the use of assurance cases and providing guidance for their use. In response, infusion pump manufacturers are beginning to use assurance cases. The FDA is the only official agency of the U.S. government that has formally mandated the use of assurance cases to date. At this time, the DoD has not yet embraced the use of assurance cases, but continuing work is focused on creating a theory of argumentation that can be used to reason about the amount of confidence in a claim that is provided by particular pieces of evidence. The expectation is that this work will lead to the ability to determine how to more effectively use scarce assurance resources.

As DoD software challenges evolve, the SEI will continue to investigate evolving engineering methods that offer promise for improving capabilities for the future. Just as assurance cases have not yet matured to the point where DoD programs are ready to apply them, they offer opportunity for improved safety. Application of such techniques in non-defense systems that exhibit more constrained characteristics provide the SEI an opportunity to demonstrate the efficacy of such engineering approaches and evolve them to be more robust.

### 6.0.8 References

[Brown 1993] Brown, Alan W.; Carney, David J.; Feiler, Peter H.; Oberndorf, Patricia A.; & Zelkowitz, Marvin V. "A Project Support Environment Reference Model," 82-89. *Proceedings of TRI-Ada Conference*, Seattle, WA, September 18-23, 1993. ACM 1993.

[Buxton1981] Buxton, John & Druffel, Larry. "Requirements for an Ada Programming Environment: Rationale for Stoneman." *Proceedings of IEEE COMPSAC. IEEE Annual International Computer and Software Conference (COMPSAC)*. San Francisco, CA, October 29-31, 1980. Institute of Electrical and Electronics Engineers, 1981.

[Druffel 1983] Druffel, Larry; Redwine, Samuel, Jr.; & Riddle, William. "The DoD STARS Program." *IEEE Computer* (November 1983): 21-30.

[DSB 1994] Defense Science Board Task Force (Larry Druffel & George H. Heilmeier, co-chairs). *Acquiring Defense Software Commercially.* (Report #859), June 1994. http://www.dod.mil/pubs/foi/logistics_material_readiness/acq_bud_fin/859.pdf

[IBM 2010] IBM. *Rational Unified Process® (RUP®) Plug-ins for Rational Method Composer 7.5.* http://www-01.ibm.com/support/docview.wss?uid=swg24028579

[Weinstock 2009] Weinstock, Charles & Goodenough, John. *Towards an Assurance Case Practice for Medical Devices* (CMU/SEI-2009-TN-018). Software Engineering Institute, Carnegie Mellon University, 2009. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=8999

## 6.1 Configuration Management

### 6.1.1 The Challenge: Configuration Support for Software Developers

When the SEI was founded in the mid-1980s, software development was already a complex activity, with single projects often involving millions of lines of code and large (often distributed) teams of developers. At the time, developers in government and industry, including government contractors, adapted established configuration management (CM) practices in other engineering disciplines, such as versioning of design documents and physical system parts and managing changes through change control boards [IEEE 1987]. They complemented these practices with version control systems to manage versions and revisions of their source code; they used databases for tracking versions and changes to the artifacts. Many were home-grown systems. However, software development presented several unique challenges. First, software development is a continuous design activity in which bug removal affects the design. Second, software design and source code are easily changed, and configuration, parameterization, and deployment of software on hardware can result in unexpected system behavior. Third, software development is easily distributed across multiple teams, thanks to networked computing environments, which were emerging in the early 1980s. Finally, at that time, system engineering drove the development process, with software developers getting involved late in the process and having to write code against ambiguous specifications and repeatedly changing requirements. (These were some of the factors contributing to the software crisis that prompted the establishment of the SEI.)

### 6.1.2 A Solution: Configuration Management Tools

In the 1980s, DARPA—the main SEI sponsor—was interested in research into configuration support for software development environments, both for Ada specifically and for environments in general. At the SEI, work was soon under way to establish a framework for improving the software process, which became known as the Capability Maturity Model [Humphrey 1988]. This framework included configuration management as a key process area, leveraging established CM practices [IEEE 1987].

From the mid-1980s to 1993, the SEI worked on tool-based solutions supporting the full software development process and complementing the organizational process focus of the CMM. The SEI quickly expanded from evaluating Ada environments [Weiderman 1987] to integrated software development environments (ISDE). Recognizing the challenges of configuration management for software, in 1988 the SEI established a series of international software configuration management workshops under the auspices of the ACM, chairing the first set of workshops [Winkler 1988, Feiler 1991]. These workshops continued for more than a decade. The SEI published papers on the role of CM in integrated environments [Dart 1989], the different roles CM plays throughout the software system lifecycle and the concepts supporting them [Feiler 1988], and process support through CM [Feiler 1989]. To facilitate collaboration between the software process and the ISDE research communities, the SEI published a set of software process development and enactment concepts and definitions [Feiler 1992]. As promising research results emerged, the SEI assessed the state of the art in ISDE and its support for CM [Brown 1992, 1993a]. In the early 1990s, the SEI provided the co-chair of the Navy NGCR Project Support Environments Standards Working Group (PSESWG) and took on a lead role in the development of a reference model for Integrated Software Engineering Environments [Brown 1993b], which included configuration management

as a key service and became a NIST/European Computer Manufacturers Association (NIST/ECMA) standard [NIST 1994].

Research into architecture modeling languages for embedded software systems and increased interest in model-based software engineering spawned new interest in CM [Westfechtel 2003, Estublier 2005]. In the mid-1990s, the SEI developed the Simplex architecture [Sha 1996], which provides software fault-tolerance for control systems through self-adaptive semantic redundancy. In the context of the Simplex architecture, the SEI worked with Carnegie Mellon University researchers to investigate the use of architecture models to analyze system configurations for inconsistencies and, during operation, to manage dynamically reconfiguring systems against known consistency constraints [Feiler 1998]. The investigation demonstrated the feasibility of extending configuration consistency into the operational environment through formalized specification and analysis of system models.

**The View from Others**

*The SEI's background and expertise were key to the development of the PSEWG Reference Model. Without these contributions, this reference model would not have been as rich or meaningful.*

– Patricia Oberndorf, U.S. Navy, Next Generation Computer Resources Program

As DARPA-funded research in architecture languages produced promising results, the SEI, in collaboration with the U.S. Army Aviation and Missile Research Development and Engineering Center *(*AMRDEC*)*, took on the technical leadership in the development of the industry standard Architecture Analysis and Design Language (AADL). AADL has been chosen as a key technology by the aerospace industry in its System Architecture Virtual Integration (SAVI) initiative because of AADL's ability to support large-scale, multi-team modeling and analysis.

### 6.1.3 The Consequence: Configuration Management and CM Tools in Common Practice

Software developers gained control over the versions and configurations during the software development lifecycle. Commercial and open source versions of configuration management tools have become an integral part of their development environment, transparent and requiring no overhead to use. Their capabilities have been extended to uniformly support individual developers' workspaces and cooperative team development. The tools also have been extended to support build and release management of artifacts ranging from documents to models, source code, binaries, build-and-installation configuration files, and other artifacts. De facto open source standards in integrated development environments (Eclipse) and distributed configuration management (GIT) have been embraced by industry and government and have been used by the SEI as the basis for OSATE.CM has become well established in the community in other ways. See, for example, Crossroads web-based resources on CM[50] and CM tool recommendations [Burrows 2005].

Despite these advances, new challenges are being posed to CM through the emergence of ultra-large-scale systems [Northrop 2006], such as web-enabled, rapidly evolving, user-adaptable systems.

50 Home page: http://www.cmcrossroads.com/

### 6.1.4 The SEI Contribution

The SEI recognized the need to complement the process focus of the CMM with a better understanding of how to develop integrated software development environments that support both individual developers and teams throughout the lifecycle. Toward that end, the SEI fostered research into configuration management concepts both that meet the needs of software configuration management users and that develop a standard reference model for integrated environments. This research resulted in a crop of CM tools that more transparently integrated with tool environments. The SEI continued to promote CM concepts as the software engineering community embraced model-based engineering and architecture-centric engineering. By taking on technical leadership of the SAE AADL standard, the SEI was able to integrate two DARPA-funded research architecture languages (MetaH and ACME) and incorporate CM support to meet practitioners' needs. By choosing AADL as a key technology in the SAVI initiative and collaborating with the SEI in establishing requirements for a model repository that include consistent CM of models, the aerospace industry has shown its appreciation of SEI's expertise in CM as well as architecture-centric, model-based engineering technologies the SEI has developed.

CM has become part of the SEI Product Line Practice Framework.

### 6.1.5 References

[Burrows 2005] Burrows, Clive & Wesley, Ian. *Ovum Evaluates: Configuration Management*. Ovum Ltd., 2005.

[Brown 1992] Brown, A.; Dart, S.; Feiler, P.; & Wallnau, K. "The State of Automated Configuration Management." *Software Engineering Institute Annual Technology Review*. Software Engineering Institute, Carnegie Mellon University, 1992. ftp://ftp.sei.cmu.edu/pub/case-env/config_mgt/papers/atr_cm_state.pdf

[Brown 1993a] Brown, A. W.; Wallnau, K. C.; & Feiler, P. H. "Understanding Integration in a Software Development Environment: Issues and Illustrations." *Journal of Systems Integration*, September 1993.

[Brown 1993b] Brown, Alan W.; Carney, David J.; Feiler, Peter H.; Oberndorf, Patricia A.; & Zelkowitz, Marvin V. "A Project Support Environment Reference Model," 82-89. *Proceedings of TRI-Ada Conference*, Seattle, WA, September 18–23, 1993. ACM 1993.

[Dart 1989] Dart, Susan A. & Feiler, Peter H. "Configuration Management in CASE Tools and Environments." *Third International Workshop on Computer-Aided Software Engineering. CASE 89*, London, July 17–21, 1989. IEEE Computer Society, 1989.

[Estublier 2005] Estublier, Jacky; Leblang, David; van der Hoek, Andre; Conradi, Reidar; Clemm, Geoffrey; Tichy, Walter; & Wiborg-Weber, Darcy. "Impact of Software Engineering Research on the Practice of Software Configuration Management." *ACM Transactions on Software Engineering and Methodology 14*, 4 (October 2005): 383-430.

[Feiler 1988] Feiler, Peter H.; Dart, Susan A.; & Downey, Grace. *Evaluation of the Rational Environment* (CMU/SEI-88-TR-015). Software Engineering Institute, Carnegie Mellon University, 1988. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=10637

[Feiler 1989] Feiler, Peter H. “Software Process Support Through Software Configuration Management.” *Fifth International Software Process Workshop.* Kennebunkport, ME, October 10–13, 1989. ACM Press, 1989.

[Feiler 1991] Feiler, Peter H., ed. *Proceedings of 3rd International Software Configuration Management Workshop.* Trondheim, Norway, June 12–14, 1991. ACM Press, 1991.

[Feiler 1992] Feiler, Peter H. & Humphrey, Watts. “Software Process Development and Enactment: Concepts and Definitions.” *Communication of the ACM 35*, 9 (September 1992): 75-90.

[Feiler 1998] Feiler, Peter H. & Li, Jun. “Consistency in Dynamic Reconfiguration.” *Proceedings of the 4th International Conference on Configurable Distributed Systems*. Annapolis, MD, May 4–6, 1998. IEEE Computer Society Press, 1998.

[Humphrey 1988] Humphrey, Watts S. “Characterizing the Software Process: A Maturity Framework.” *IEEE Software 5*, 2 (March 1988): 73-79.

[IEEE 1987] Institute of Electrical and Electronic Engineers. *IEEE Guide to Software Configuration Management*. (IEEE/ANSI Standard 1042-1987). IEEE, 1987.

[NIST 1994] National Institute of Standards and Technology. *Reference Model for Project Support Environments*. NIST Special Publication 500-213. (Co-published as Technical Report ECMA TR/69). NIST, 1994.

[Northrop 2006] Northrop, L.; Balzer, R.; Sullivan, K.; Gabriel, R.; Smith, D.; Klein, M.; Feiler, P., et al. *Ultra-Large-Scale Systems: Software Challenge of the Future*. Study and report commissioned by Claude M. Bolton Jr., Assistant Secretary of the Army (Acquisition, Logistics, and Technology). Software Engineering Institute, Carnegie Mellon University, 2006. http://www.sei.cmu.edu/uls/

[Sha 1996] Sha, R. Rajkumar & Gagliardi, M. “Evolving Dependable Real-Time Systems,” 335-346. *1996 IEEE Aerospace Applications Conference Proceedings.* Aspen, CO, March 10, 1996. IEEE, 1996.

[Weiderman 1987] Weiderman, Nelson H. *Evaluation of Ada Environments: Executive Summary* (CMU/SEI-87-TR-1). Software Engineering Institute, Carnegie Mellon University, 1987. http://repository.cmu.edu/cgi/viewcontent.cgi?article=1139&context=sei

[Westfechtel 2003] Westfechtel, Bernhard & Conradi, Reidar. “Software Architecture and Software Configuration Management,” 24-39. *Proceedings of the ICSE Workshops SCM 2001 and SCM 2003: Selected Papers*. van der Hoek, A. & Westfechtel, B., eds. (Published as Lecture Notes in Computer Science 2649). Springer-Verlag, 2003.

[Winkler 1988] Winkler, J. F. H., ed. *Proceedings of the International Workshop on Software Version and Configuration Control*. Grassau, Germany, January 27-29, 1988. ACM Press, 1988.

## 6.2 CASE Environments

### 6.2.1 The Challenge: Making Smart Decision on Tools and Environments

In the late 1980s, a number of computer-aided software engineering (CASE) tools[51] had become available with claims about how they could provide benefits for developing better software. Expectations were high for these CASE tools, particularly those that supported modeling of software, but also for new generations of configuration management/version control, code analysis, testing, and other tools. DoD program managers expressed a strong need for help in making decisions on the competing claims of tool vendors.[52]

### 6.2.2 A Solution: CASE Tool Integration

The SEI responded to the DoD need in 1989 by analyzing how to best help DoD organizations make better decisions on the selection, adoption, and integration of CASE tools. In addition, the SEI convened a series of workshops to better understand the requirements of DoD managers, as well as to get the perspectives of the research community and tool-vendor community [Huff 1992a, 1992b].

Although there had initially been sentiment from some DoD programs to develop a rating scale for the various tools, the results of the workshops and the SEI team's ongoing analysis indicated that this approach was infeasible. It would be difficult to procure the many tools entering the marketplace, construct the necessary computing environment, install the tools, and train staff in their use. In addition, because the CASE tools evolve over time, information provided for one tool version could be invalidated by the next version.

As a result, the SEI addressed DoD needs by focusing initially on strategies for adopting and integrating a set of tools, and it developed a widely used *Guide to CASE Adoption* [Smith 1992]. This guide emphasized the need for making an informed decision by identifying a need selection criteria, performing trial implementations, and defining an adoption strategy. The SEI addressed issues of cost, performance, process support, maintenance, data management, tool integration, and standardization [Zarrella 1991]. In a series of case studies, the SEI found that the state of the practice of CASE tool use was modest compared to the marketing claims [Rader 1993]. However, these studies documented the ways in which organizations overcame the shortcomings of tools current at that time through commitment, ingenuity, and attention to end-user needs.

The approach to tool integration evolved. The commercial CASE market had initially focused on vendor-centric individual tools, with a predominant emphasis on analysis and design tools. Many organizations were making improvements to their software engineering practices by incorporating various types of CASE tools, but the tools typically did not work together. As a result, manual intervention was often used to move data between tools, but this solution was both impractical and, in some cases, nearly impossible because of the divergent data models and interaction strategies of the individual tools. Some vendors positioned their tool as an integration platform, thereby locking users into the specific tool, associated development processes, and related vendors. (There

51 Examples were CADRE Teamwork and Software Through Pictures.

52 The SEI Senior Technical Review Committee expressed this as one of the top priorities from the perspective of DoD programs.

were a few exceptions. For instance, the Rational Environment was a collection of integrated tools, although it was limited to supporting Ada.)

Government- and association-sponsored efforts, rather than focusing on a specific vendor product or product suite, developed standards for open-tool-integration frameworks that could incorporate a variety of tools and reflect many processes. Extensive projects were developed in the United States, with work on the Common APSE Interface Set—Revision A [CAIS-A 1986]; in Europe, ECMA [ECMA 1997] defined the standards for the Portable Common Tool Environment (PCTE).

The SEI gathered insights on how organizations were integrating tools in practice—the software and hardware environments in which tool integration occurred, the goals of integration, the tools integrated, mechanisms used, and the standards applied [Morris 1991]. This data on the state of tool and environment integration at that time, as well as emerging trends in integration, were valuable starting points for experiments in integration that the SEI undertook. The SEI identified the potential importance of message passing as an integration mechanism [Brown 1992] and discussed the use of such a mechanism as the basis for a more flexible environment that is open to experimentation with different approaches to integration. This discussion led to a series of tool-integration experiments [Zarrella 1994]. These experiments integrated a set of CASE tools using a combination of data-integration mechanisms (PCTE, Object Management System [OMS] and UNIX file system) and control-integration mechanisms (Broadcast Message Server [BMS] of HP SoftBench). The experiments demonstrated the possibility of integrating CASE tools using a message-passing approach that is independent of the integration-framework product used. The work on CASE integration culminated in the book *Principles of CASE Tool Integration* [Brown 1994].

The National Security Agency funded the CASE environments SEI work for seven years and provided four resident affiliates for the SEI team. The team also supplied two members to the Navy's NGCR-PSEWG. These members co-authored a reference model for project-support environments [Brown 1993].

### 6.2.3 The Consequence: CASE Tools Widely Used in Practice

The CASE adoption work was widely used within DoD organizations and commercial organizations. The SEI *Guide to Case Adoption* served as the starting point for IEEE [IEEE 1995] and, later, ISO recommended practices on CASE adoption [ISO/IEC 2007]. SEI staff members were invited to be co-editors of these reports.

The work on integration had an immediate impact on significant DoD and Federal Aviation Administration acquisitions. The SEI experiments that demonstrated the utility of control integration through message passing helped to influence tool vendors and researchers to look at this technique as a more fruitful approach to interoperability than the earlier emphases on data integration.

With the SEI reputation of giving informed, unbiased technical judgments, its expertise was sought for complex decision making by program managers who had to make decisions on major acquisitions, such as the I-CASE procurement, FAA in-flight system, and the Ballistic Missile Defense Organization. In addition, its expertise was requested for DARPA research and demonstration projects, including STARS and Evolutionary Design of Complex Software (EDCS). The NSA used the insights from the *Guide to CASE Adoption* as well as insights from the integration experiments extensively in the adoption of its own software tool environment.

### 6.2.4 The SEI Contribution

The CASE work demonstrated how a small group of technical people from the SEI were able both to influence significant DoD programs and to identify challenges that had significant influence on the research community. At the time, many program managers were focused on the question, "Which is the best tool?" The SEI stepped back to ask, "What is the underlying problem that this question is trying to solve?" The work on CASE adoption provided valuable guidance for DoD program managers. It was also used as an example of best practice in the work of the IEEE and ISO/IEC recommended practices.

The work on integration had an immediate impact on DoD and FAA acquisitions, along with tool adoption in other government agencies. The thread was carried though in later research on SOA and COTS-based integration. It influenced both the broader research community as well as later SEI efforts in COTS-based systems, predictable assembly from certifiable components (PACC), integration of software-intensive systems, systems of systems, and advanced mobile systems.

### 6.2.5 References

[Brown 1992] Brown, Alan. *Control Integration through Message Passing* (CMU/SEI-92-TR-035). Software Engineering Institute, Carnegie Mellon University, 1992. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=11757

[Brown 1993] Brown, Alan W.; Carney, David J.; Feiler, Peter H.; Oberndorf, Patricia A.; & Zelkowitz, Marvin V. "A Project Support Environment Reference Model," 82-89. *Proceedings of TRI-Ada Conference*. Seattle, WA, September 18–23, 1993. ACM 1993.

[Brown 1994] Brown, A. W.; Carney, D. J; Morris, E. J.; Smith, D.; & Zarrella, P. *Principles of CASE Tool Integration*. Oxford University Press, 1994 (ISBN 0195094786).

[CAIS-A 1986] Munck, R; Oberndorf, P; Ploedereder, E; & Thai, R. *An Overview of DOD-STD-1838A (proposed), The Common APSE Interface Set, A Revision*. Department of Defense, 1986.

[ECMA 1997] ECMA. *Standard ECMA-149 4th Edition*. Portable Common Tool Environment (PCTE)—Abstract Specification. ECMA, 1997.

[Huff 1992a] Huff, Cliff; Smith, Dennis; Stepien-Oakes, Kimberly; & Morris, Edwin. *Proceedings of the CASE Adoption Workshop* (CMU/SEI-91-TR-014). Software Engineering Institute, Carnegie Mellon University, 1992. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=11385

[Huff 1992b] Huff, Cliff; Smith, Dennis; Morris, Edwin; & Zarrella, Paul. *Proceedings of the CASE Management Workshop* (CMU/SEI-92-TR-006). Software Engineering Institute, Carnegie Mellon University, 1992. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=11601

[IEEE 1995] Institute of Electrical and Electronic Engineers. *IEEE Recommended Practice for the Adoption of Computer-Aided Software Engineering (CASE) Tools.* (IEEE Standard1348-1995). IEEE, 1995. http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?reload=true&punumber=3688

[ISO/IEC 2007] ISO/IEC/JTC 1/SC 4. *Guidelines for the Adoption of CASE Tools* (TR 14471). ISO/IEC, 1997.

[Morris 1991] Morris, Ed; Feiler, Peter; & Smith, Dennis. *CASE Studies in Environment Integration* (CMU/SEI-91-TR-013). Software Engineering Institute, Carnegie Mellon University, 1991. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=11381

[Rader 1993] Rader, Jock; Brown, Alan; & Morris, Edwin. *An Investigation into the State of the Practice of CASE Tool Integration* (CMU/SEI-93-TR-015). Software Engineering Institute, Carnegie Mellon University, 1993. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=11909

[Smith 1992] Smith, Dennis; Morris, Edwin; & Stepien-Oakes, Kimberly. *Guide to CASE Adoption* (CMU/SEI-92-TR-015). Software Engineering Institute, Carnegie Mellon University, 1992. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=11647

[Zarrella 1991] Zarrella, Paul; Smith, Dennis; & Morris, Edwin. *Issues in Tool Acquisition* (CMU/SEI-91-TR-008). Software Engineering Institute, Carnegie Mellon University, 1991. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=11349

[Zarrella 1994] Zarrella, Paul & Brown, Alan. *Replacing the Message Service Component in an Integration Framework* (CMU/SEI-94-TR-017). Software Engineering Institute, Carnegie Mellon University, 1994. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=12217

## 6.3 Software Technology Reference Guide

### 6.3.1 The Challenge: Effective Software Technology Adoption

Software technology adoption is a challenge in a field where the technology is constantly changing, along with the needs of the adopters. Technology consists "not just of the technical artifacts but the knowledge embedded in those artifacts and the knowledge required for their effective use" [Foreman 1997, p. 44]. Technology users need knowledge that enables them to systematically plan research and development (R&D), as well as perform technology insertion activities to meet their organization's current and future needs.

### 6.3.2 A Solution: Software Technology Reference Guide

The U.S. Air Force acquisition community tasked the SEI to create a reference document that would provide the Air Force with greater understanding of software technologies to support its R&D and adoption plans.

Members of the SEI, the Air Force,[53] and government contractors worked as a cooperative team to produce the *Software Technology Reference Guide* (STRG), which was first published in 1997 [Foreman 1997]. In a rather novel approach, several government contractors[54] provided personnel for several months to aid in the evolution of document concepts and to author the majority of the technology descriptions. In the early phases of the project, the document was also referred to as the Software Technology Roadmap and the Structured Survey of Software Technology.

The 1997 reference guide included then-current information on 60 technologies, each described in four to six pages. The descriptions underwent rigorous review by nearly 50 experts in the community. The guide emphasized software technologies that were important to the command, control, communications, computers, and intelligence (C4I) area; however, much of the information could be applied broadly. The information was relevant to any complex, large-scale, distributed, real-time, software-intensive, embedded system. The major concerns for these systems are reliability, availability, safety, security, performance, maintainability, and cost. In 1998, the STRG was "reengineered," becoming one of the first interactive, web-enabled reference guides, which significantly increased its availability and impact.

### 6.3.3 The Consequence: Unbiased Information Used for Selecting Technology

The *Software Technology Reference Guide* provided common ground for contractors, researchers, government program offices, and other software-related organizations to assess technology. The information in the guide was encapsulated so that readers could rapidly make a preliminary decision as to whether further study/examination of a technology for potential use was warranted. The technology descriptions layered information so that readers could get a focused synopsis of the technology and find subordinate technology descriptions and pointers to sources of more detailed information, including the experience of others. In addition, the technology descriptions provided

53 Capt Mark Gerken and Elizabeth Kean, Rome Laboratory; Capt Gary Haines, AFMC SSSG; and Maj David Luginbuhl, Air Force Office of Scientific Research [Foreman 1997].

54 Lockheed Martin (Michael Bray and William Mills); GTE (Darleen Sadoski); E-System (James Shimp); Kaman Sciences (Edmond Van Doren); and TRW (Cory Vondrak) [Foreman 1997].

insight into costs, risks, quality, ease of use, and alternatives; it described the shortcomings of technologies as well as the advantages. Thus, readers knew where to go for more information and what questions to ask.

Readers gained knowledge that allowed them to make well-informed plans and decide on the best route to selecting and adopting technology that met their particular needs.

### 6.3.4 The SEI Contribution

Developing the *Software Technology Reference Guide* was a cooperative effort of the SEI, defense contractors, and the U.S. Air Force. The SEI members of the development team established the project direction, evolved the template of the technology descriptions, wrote some of the technology descriptions, integrated and edited contributions from all team members, published the document, and made it available to the public through the SEI website in both document and interactive online versions. Prior to this effort, no other source provided the same type of software technology information in one place. The knowledge provided in the guide enabled the Air Force to systematically plan the research and development and technology insertion required to meet current and future Air Force needs, from the upgrade and evolution of current systems to the development of new systems. The web-enabled version, in particular, gave the same benefits to the broader community. In fact, the STRG was the number one most referenced set of webpages at the SEI for several years.

### 6.3.5 References

[Foreman 1997] Foreman, John; Gross, Jon; Rosenstein, Robert; Fisher, David; & Brune, Kimberly. *C4 Software Technology Reference Guide: A Prototype* (CMU/SEI-97-HB-001). Software Engineering Institute, Carnegie Mellon University, 1997. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=12689

## 6.4 Reengineering

### 6.4.1 The Challenge: Legacy Software in Defense Systems

In the early 1990s, the DoD had a number of legacy systems that had been developed in the previous decades to inject automated technology into the nation's defense. At the time these systems were developed, there were no precedents or guidelines for developing, managing, or updating large-scale systems [Goodenough 1970]. However, the problem of updating and revising legacy systems was growing rapidly; it had become particularly difficult because of the sheer size and magnitude of DoD systems, as well as the fact that contracts for many of the early systems had been awarded to the lowest bidder, thereby inhibiting "extra work" to design for maintainability. As a result, there was a diversity of hardware and software in systems that were difficult to modify and to integrate with other systems.

The challenge for DoD program managers was to make both programmatic and technical decisions on whether to maintain existing systems, reengineer existing systems to insert new capabilities, or develop replacement systems. There was little guidance for making these decisions.

### 6.4.2 A Solution: A Reengineering Center

The SEI received strong statements of need from DoD stakeholders to address the problem of maintenance and reengineering. The SEI wrote a white paper on options for addressing this need [Feiler 1993]. This paper recognized that the research community had been active in addressing the reengineering problem and that reengineering did not represent a novel technical problem; its solution required the application of known software engineering principles to a different problem set. A seminal paper defining the field of reengineering had been published [Chikofsky 1990], and extensive research was underway; for example, a comprehensive set of papers and articles had recently been published [Arnold 1993]. Consequently, the paper recommended that the SEI become a clearinghouse of research for the DoD and research community, rather than initiate a new research program that could duplicate current work. A Reengineering Center was established with a small group of people to encourage and leverage work within the SEI, serve as a conduit between SEI work and the research community, communicate DoD needs to the research community, and make research findings accessible to the DoD. In this role, the SEI would become part of the community, articulate major challenges that needed to be addressed—especially those that were being faced by DoD program managers, and leverage existing resources and research. It would also fill the important role of enabling DoD organizations to separate reality from the large volume of claims that were permeating the media.

The SEI technical approach encompassed three broad areas:

1. development of broad frameworks that articulated DoD needs for the research community through a collection of papers. One set of papers provided a framework for program understanding, which was a key technical issue for software reengineering [Tilley 1996a]. Another set identified trends and needs and provided a focus for research in reengineering and related areas [Tilley 1996b]. A third set focused on software migration, and this was used by DoD programs in making comprehensive migration decisions [Bergey 2001].
2. publication of online resources of reengineering materials that included an identification and classification of challenges, as well as resources DoD program managers could use in making decisions [Bergey 1999, Feiler 1993, Kontogiannis 2009].

3. leveraging of other SEI work from related technical areas and work from the reengineering community to provide support for DoD programs [Kazman 1999]. This work included the areas of architecture reconstruction and architecture models for reengineering [Kazman 1998].

### 6.4.3 The Consequence: Effective Decision Making About Reengineering

The Army Training Support Center (ASTC), FAA, and Joint Logistics Center (JLC), among other organizations, used SEI support for their reengineering programs. At ATSC, a model was implemented for identifying and providing guidance for making reengineering decisions at key points in the program lifecycle. The ATSC used the migration framework as a framework for making decisions on migration to a family of new systems, and the National Security Agency used it as a basis for migrating to a set of reengineering tools and methods. The JLC work resulted in guidance for choosing methods and technology that was used widely within the DoD community. The effort with FAA resulted in a series of recommendations that were followed rigorously in the implementation of the currently operational FAA in-flight system.

### 6.4.4 The SEI Contribution

The SEI researchers identified broad trends and needs in reengineering for the DoD. As a result of the frameworks that were developed, the SEI established credibility and was invited to participate in leadership roles in the broader research community. The SEI focused on several areas that were of direct importance to the DoD, such as developing a migration model and a model for program understanding. A clearinghouse of reengineering information was developed and distributed on the SEI website in order to make it available for DoD programs. The SEI provided direct support for the DoD and other federal government programs, including the FAA, Army ATSC, JLC, and NSA.

DoD organizations recognized the leadership role of the SEI in reengineering and migration of legacy assets. The DoD JLC requested SEI technical participation on its Software Reuse Committee. The SEI project leader was invited to lead workshops on software reengineering challenges for the DoD and to participate in an updated version of the DoD software reengineering handbook.

The SEI technical papers contributed to recognition of SEI researchers as leaders in the field; SEI staff members presented papers, workshops and tutorials at conferences in the area. They were also elected to leadership positions in conferences including IWPC, STEP and ICSM.

The Reengineering Center formalized the SEI's role as a link between DoD needs, other SEI research, and the external research community. It established mechanisms that would be replicated in other areas. The approach of identifying open research challenges, becoming leaders within a specific research community, serving as a broker between DoD needs and the research community, identifying specific technical gaps for SEI research, and providing direct support to selected programs, is not unique to the reengineering work. This approach has evolved to become standard in most current SEI research projects and has been adopted by SEI technical projects as well.

### 6.4.5 References

[Arnold 1993] Arnold, Robert S. *Software Reengineering*. IEEE Computer Society Press, 1993 (ISBN 0818632720).

[Bergey 1999] Bergey, John; Smith, Dennis; Tilley, Scott; Weiderman, Nelson; & Woods, Steve. *Why Reengineering Projects Fail* (CMU/SEI-99-TR-010). Software Engineering Institute, Carnegie Mellon University, 1999. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=13405

[Bergey 2001] Bergey, John; O'Brien, Liam; & Smith, Dennis. *DoD Software Migration Planning* (CMU/SEI-2001-TN-012). Software Engineering Institute, Carnegie Mellon University, 2001. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=5489

[Chikofsky 1990] Chikofsky, E. & Cross, J. "Reverse Engineering and Design Recovery: A Taxonomy." *IEEE Software 7*, 1(January 1990):13-18.

[Feiler 1993] Feiler, Peter. *Reengineering: An Engineering Problem* (CMU/SEI-03-SR-5). Software Engineering Institute, Carnegie Mellon University, 1993. http://resources.sei.cmu.edu/asset_files/SpecialReport/1993_003_001_16139.pdf

[Goodenough 1970] Goodenough, John. Appendix A in Overton, R. K. et al. *A Study of the Fundamental Factors Underlying Software Maintenance Problems: Final Report* (ESD-TR-72-121). Deputy for Command and Management Systems, HQ Electronic Systems Division (AFSC), 1971.

[Kazman 1998] Kazman, R.; Woods, S.; & Carrière, J. "Requirements for Integrating Software Architecture and Reengineering Models: CORUM II," 54-153. *Proceedings of the Fifth Working Conference on Reverse Engineering (WCRE)*. Honolulu, HI, October 12–14, 1998. IEEE Computer Society Press, 1998.

[Kazman 1999] Kazman, R. & Carrière, J. "Playing Detective: Reconstructing Software Architecture from Available Evidence." *Automated Software Engineering 6*, 2 (April 1999): 106-138.

[Kontogiannis 2009] Kontogiannis, Kostas; Lewis, Grace; & Smith, Dennis B. *A Research Perspective on Maintenance and Reengineering of Service-Oriented Systems.* Software Engineering Institute, Carnegie Mellon University, 2009. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=29035

[Tilley 1996a] Tilley, S. R.; Paul, S.; & Smith, D. B. "Towards a Framework for Program Understanding," 19-28. *Proceedings of Fourth Workshop on Program Comprehension.* Detroit, MI, March 29–31, 1996. IEEE, 1996.

[Tilley 1996b] Tilley, Scott & Smith, Dennis. *Coming Attractions in Program Understanding* (CMU/SEI-96-TR-019). Software Engineering Institute, Carnegie Mellon University, 1996. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=12613

## 6.5 Building and Fielding Interoperating Systems

### 6.5.1 The Challenge: Interoperability in Evolving Defense Systems

Most of the software-reliant systems that traditionally supported operations in the DoD, particularly in the military theater—including planning, manpower, and logistics—were designed as stand-alone systems. They had limited capabilities for net-centric warfare, such as the efficient and secure exchange of information across a networked force of weapons, sensors, and soldiers. In addition, acquisition practices, system architectures, and engineering solutions were optimized for stand-alone systems. The Gulf War I (circa 1990) catalyzed a new emphasis on the timely, efficient, and secure exchange of information across independently developed software-reliant systems, with interoperation among them—a particularly significant need because of the war's multi-service nature. This critical need was not just in the DoD and with its coalition partners, but in civilian government and industry as well.

Many organizations recognized that a system of systems (SoS)[55] could help them achieve important business and mission goals more effectively; however, they needed guidance on how to create systems of systems out of legacy systems and new systems (either fielded or under development) that were not necessarily designed to work together. The challenge is significant because organizations must create systems of systems when it is difficult to know which systems will need to be involved in the future and when knowledge of system behavior is incomplete. It is equally difficult to predict what data a system holds that will be of value to others. Finally, in a system of systems, there is often no practical central control over all the systems involved. The challenge is complex, with technical, operational, governance, management, and acquisition aspects.

### 6.5.2 A Solution: Multi-Faceted Approach to Support for Interoperation

The SEI started, in the mid-1990s, to seek solutions to the interoperation of independently evolving systems, focusing on integrating commercial off-the-shelf products. In 2002–2003, the SEI conducted the System of Systems Interoperability (SOSI) research project. Through this project, the researchers realized that interoperability must occur not only at the technical level but also at the program office/governance level. By 2004, the SEI's focus shifted to finding a general solution for the interoperability needs of government and industry organizations as they faced unprecedented issues in migrating to network-centric operations and systems of systems.

The approach to systems of systems was two pronged: defining fundamental principles and concepts and developing techniques for putting the principles into practice. Among the principles were these:

1. Independent and continuous evolution of individual systems means that system-of-systems behavior cannot be completely predicted.

55 Admiral William Owens introduced the concept of a system of systems in a 1996 paper, defining it as the serendipitous evolution of a system of intelligence sensors, command and control systems, and precision weapons that enabled enhanced situational awareness, rapid target assessment, and distributed weapon assignment [Owens 1996]. The term *system of systems* is now used in a variety of settings, beyond those in the original definition.

2. Influence is more important than control in achieving collaboration among system-of-systems stakeholders, as a single point of control over the various individual systems is not practical.

In communicating principles and issues, the SEI also raised awareness of the complexity of the problem—however simply solutions could be depicted on paper, they were not simple to implement [Brownsword 2004, Fisher 2006, 2007]. While the principles helped to characterize the key differences of systems of systems, they were not sufficient to determine needed governance, acquisition, and engineering practices. In concert, the SEI leveraged insights from Boxer's "Double Challenge" of relating who must collaborate in the SoS effort with what each must provide [Anderson 2007]. With increased complexity in the number and interactions of the enterprises that must collaborate, along with the need for solutions that address dynamically changing operational needs—which often occur in unanticipated ways—traditional governance and engineering practices alone were no longer viable.

For the governance and acquisition aspect, the SEI looked at the relationships and the influence and reward structures in SoS collaborations, ensuring relevant stakeholder participation in decision-making and providing guidance on SoS acquisition and evolution. SEI experts developed approaches such as SoS Navigator [Boxer 2008]. Navigator takes a social-technical view of systems of systems, addressing technical, business, and people aspects, particularly in dynamic demand environments. It enables leaders to address critical aspects of their organizations' demand and supply sides and decide whether to adopt a different business model. The Navigator approach offers tools and techniques, such as Critical Context Analysis (originally called Collaboration Stakeholder Analysis), and identifies key implications to processes, interoperability, engineering, and collaboration.

On the engineering side, the SEI developed lifecycle guidelines and tutorials [Lewis 2008a, 2008b; Smith 2008] for engineering practices such as requirements engineering, testing, maintenance, and evolution. To gain experience with engineering techniques where capabilities are under the control of different organizations, the SEI leveraged its research with service-oriented architecture, an architectural style defined by the OASIS Reference Model as "a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains" [OASIS 2006]. SEI SoS engineering techniques include SMART (SOA Migration,

**The View from Others**

*The SEI team gave us a new way to look at the breadth of the system-of-systems environment, and we now understand why it is so hard.*

– Attendees of the SEI's National Defense Industrial Association tutorial [Garcia-Miller 2009]

*Based on the engineering principle of loose coupling, SOAs manage software system interactions using standardized interfaces. Using a services-oriented approach helps move from a set of interlocked, point-to-point interfaces to more effective means of interoperability and data sharing… We expect SMART to help us conduct the kind of rigorous analysis that allows us to make the best decisions.*

– Dr. Tim Rudolph, Chief Technology Officer, U.S. Air Force Electronic Systems Center [Paone 2009]

Adoption, and Reuse Technique), which is a family of techniques[56] that helps organizations migrating legacy applications to a service-oriented architecture by making better decisions about SOA adoption [SEI 2013, Lewis 2008b].

### 6.5.3 The Consequence: Well-Informed Decisions Using Tools and Techniques

The SEI has provided to the DoD and defense contractors tools and techniques for making well-informed decisions about systems of systems based on data, both quantitative and qualitative. They are better able to integrate additional, separately developed systems into their systems of systems, even with incomplete technical, operational, and business information. They are aware that, for systems of systems, the organizational impact is an essential element of system design and deployment. They are aware that their acquisition approach must mirror a realistic view of software and systems engineering when interoperability is a high priority. They have principles and techniques for determining the feasibility of and building a plan for moving to a system-of-systems environment. They have the additional benefit of SEI case studies that illustrate, for example, decisions by government organizations on the interoperability of systems and whether legacy systems can be migrated to a SOA environment.

### 6.5.4 The SEI Contribution

The SEI has provided unbiased guidance and techniques to aid the successful engineering, governing, and acquisition of systems of systems. In addition, the SEI developed case studies and training courses. It applied its research findings to real-world situations through engagements with customers. Those experiences and collaborations with other researchers enabled SEI experts to develop, pilot, mature, and make available system-of-systems tools and methods for migrating new and legacy systems to system-of-systems environments and for evaluating the effectiveness of systems interoperability technologies. The SEI SoS work influenced products that were sponsored by the Office of the Secretary of Defense [OSD 2008]. The early SEI work on SoS principles and concepts included defining characteristics of a system of systems. Here the SEI adapted work by Maier [Maier 1998] and others, such as White [White 2005].

### 6.5.5 References

[Anderson 2007] Anderson, Bill; Boxer, Philip J.; Morris, Edwin J.; & Smith, Dennis B. “The Double Challenge in Engineering Complex Systems of Systems.” *News at SEI*. Software Engineering Institute, Carnegie Mellon University, 2007. http://www.sei.cmu.edu/library/abstracts/news-at-sei/eyeonintegration200705.cfm

[Boxer 2008] Boxer, Philip; Carney, David; Garcia-Miller, Suzanne; Brownsword, Lisa; Anderson, William; Kirwan, Patrick; Smith, Dennis; & Morley, John. *SoS Navigator 2.0: A Context-Based Approach to System-of-Systems Challenges* (CMU/SEI-2008-TN-001). Software Engineering Institute, Carnegie Mellon University, 2008. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=8515

60 See http://www.sei.cmu.edu/architecture/tools/smart/index.cfm and [Lewis 2008b] for descriptions.

[Brownsword 2004] Brownsword, Lisa; Carney, David; Fisher, David; Lewis, Grace; Morris, Edwin; Place, Patrick; Smith, James; Wrage, Lutz; & Meyers, B. *Current Perspectives on Interoperability* (CMU/SEI-2004-TR-009). Software Engineering Institute, Carnegie Mellon University, 2004. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=7109

[Fisher 2006] Fisher, David. *An Emergent Perspective on Interoperation in Systems of Systems* (CMU/SEI-2006-TR-003). Software Engineering Institute, Carnegie Mellon University, 2006. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=8061

[Fisher 2007] Fisher, David; Meyers, B.; & Place, Patrick. *Conditions for Achieving Network-Centric Operations in Systems of Systems* (CMU/SEI-2007-TN-003). Software Engineering Institute, Carnegie Mellon University, 2007. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=8207

[Garcia-Miller 2009] Garcia-Miller, Suzanne; Brownsword, Lisa; & Kirwan, Patrick. "Organizational Implications of Systems of Systems" (tutorial). National Defense Industrial Association (NDIA) October 2009. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=21548

[Lewis 2008a] Lewis, G.; Morris, E.; Place, P.; Simanta, S.; Smith D.; & Wrage, L. "Tutorial: Engineering Systems of Systems." *Proceedings of the Seventh International Conference on Composition-Based Software Systems (ICCBSS)*. Madrid, Spain, February 25–29, 2008. IEEE Computer Society, 2008.

[Lewis 2008b] Lewis, Grace; Morris, Edwin; Smith, Dennis; & Simanta, Soumya. *SMART: Analyzing the Reuse Potential of Legacy Components in a Service-Oriented Architecture Environment* (CMU/SEI-2008-TN-008). Software Engineering Institute, Carnegie Mellon University, 2008. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=8571

[Maier 1998] Maier, Mark W. "Architecting Principles for Systems-of-Systems." *Systems Engineering 1*, 4 (1998): 267-284. John Wiley & Sons, 1998. http://onlinelibrary.wiley.com/doi/10.1002/%28SICI%291520-6858%281998%291:4%3C267::AID-SYS3%3E3.0.CO;2-D/abstract

[OASIS 2006] OASIS. *Reference Model for Service-Oriented Architecture1.0*. (OASIS Standard). OASIS, 2006. http://docs.oasis-open.org/soa-rm/v1.0

[OSD 2008] Office of the Deputy Under Secretary of Defense for Acquisition and Technology, Systems and Software Engineering. *Systems Engineering Guide for Systems of Systems, Version 1.0*. ODUSD(A&T)SSE, 2008. http://www.acq.osd.mil/se/docs/SE-Guide-for-SoS.pdf

[Owens 1996] Owens, William A. *The Emerging U.S. System-of-Systems*, Institute for National Strategic Studies, National Defense University (Strategic Forum 63), 1996. http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA394313

[Paone 2009] Paone, Chuck. "ESC Enters Pioneering Agreement with Software Engineering Institute." Electronic Systems Center, Hanscom Air Force Base, 2009. http://www.hanscom.af.mil/news/story.asp?id=123136939

[SEI 2013] "SOA Migration, Adoption, and Reuse Technique (SMART) Materials." Software Engineering Institute, 2013. http://www.sei.cmu.edu/architecture/tools/smart/index.cfm

[Smith 2008] Smith, D. & Lewis, G. “Systems of Systems: New Challenges for Maintenance and Evolution,” 149-157. *Proceedings of the 2008 Frontiers of Software Maintenance* (art. no. 4659258), IEEE International Conference of Software Maintenance, FoSM Program. Beijing, China, October 2008. IEEE/FoSM, 2008.

[White 2005] White, B. E. “A Complementary Approach to Enterprise Systems Engineering.” *NDIA Systems Engineering Conference*. San Diego, CA, October 24–27, 2005. The MITRE Corp., 2005. http://www.dtic.mil/ndia/2005systems/wednesday/white.pdf

## 6.6 Developing Systems with Commercial Off-the-Shelf Products

### 6.6.1 The Challenge: Using Commercial Off-the-Shelf Products in Defense Systems

Incorporating commercial components[57] into new systems has long been an effective means to save time and money in building large software systems. In 1986, the Packard Commission urged the DoD to "develop new or custom-made items only when it has been established that those readily available are clearly inadequate to meet military requirements" [Packard 1986]. In 1994, the DoD was similarly urged to use commercial off-the-shelf (COTS) products by a Defense Science Board Report, *Acquiring Defense Software Commercially* [DSB 1994]. There was early DoD concern about how COTS products could meet the rigid military security and performance requirements, and the DoD took initial steps toward the use of these products by establishing several initiatives. For example, the Navy conducted the NGCR program to identify interface standards for operating systems, networks, and several other areas as a step toward open systems, which, in turn, encouraged the development and use of COTS products.

At that time, there was substantial movement toward using COTS operating systems, but introducing the use of COTS products at the level described above proved to be challenging—and was accompanied by an increase in program failures. The increased use of COTS products meant that developers lost design control, which was now influenced by market forces. Competitive pressures in the software marketplace motivated vendors to innovate and differentiate features rather than to stabilize and standardize, making component integration difficult and increasing design complexity. Knowledge obtained about one commercial software component did not translate easily from one vendor to another and tended to degrade quickly as the products evolved through new releases. Because integrating COTS products proved to be a delicate and difficult task, specific engineering practices were needed to accomplish that task successfully [Wallnau 2001].

### 6.6.2 A Solution: Tools and Guidance for Improved Use of COTS Products

When the significant need in the COTS area became apparent, the SEI was poised to provide much-needed expertise. The SEI's CASE (computer-aided software engineering) Environments work provided a strong foundation, and the SEI staff had a breadth of related experience that could be brought to bear. In 1997, the COTS-Based Systems (CBS) initiative was created. In a typical SEI approach, the initiative developed basic principles as a foundation, and developed processes, tools, and methods for acquiring, engineering, and managing COTS-based systems. The SEI emphasized the convergence in four spheres needed for success: stakeholder needs and business processes; system architecture and design; the marketplace; and management processes, concerns, and constraints that govern the development of the system, including risk management. The keys to success from this perspective are iterative negotiation and knowledge building among these spheres, improving the basis for making decisions. The SEI worked directly with several government programs (for example, the Business Information System Program Office, Electronic

57 Examples of commercial components range from platform-level (such as HTTP servers and transaction monitors) to general purpose (such as web browsers and relational database management systems) to domain specific (such as tax preparation packages, geographic information systems, and biometric identity products).

Systems Command, and the Joint Engineering Data Management Information and Control System—JEDMICS), thereby helping them while refining its body of knowledge, processes, and tools. The resulting, government-oriented body of knowledge was reflected in training courses, technical reports, journal articles, presentations, and the coaching and advice the SEI brought to its customers.

Some of the more significant results are the following. The CBS Activity Framework [Brownsword 2000] guides program managers in activities and practices that are essential for developing and supporting COTS-based systems. The SEI also gives acquisition managers and policy makers a basic understanding of how developing systems with COTS products is different and of capabilities that can assist them [Brownsword 1998]. The Evolutionary Process for Integrating COTS-Based Systems (EPIC) [Albert 2002a, 2002b] redefines acquisition, management, and engineering practices to more effectively take advantage of the COTS marketplace and other sources of pre-existing components. To reduce the number of program failures attributable to COTS, the SEI developed the COTS Usage Risk Evaluation (CURE) and an Assembly Process for COTS-Based Systems (APCS), a process framework for developing software systems based on COTS products [Carney 2003]. The framework is based on Barry Boehm's familiar spiral development process [Boehm 1986]. In a cooperative effort, the SEI and National Research Council, Canada defined a process for evaluating COTS software products, called PECA (which stands for Plan evaluation, Establish criteria, Collect data, Analyze results) [Comella-Dorda 2004]. Because of the expanding role of CMMI across the SEI customer base, the SEI produced an interpretation of CMMI for COTS-based systems [Tyson 2003]. This interpretation solidified the basis of EPIC and related work by confirming that, although one could modify the process to suit one's needs, the *principles* could not be modified or ignored.

An outgrowth of the SEI work in COTS-based systems was an investigation into whether it is possible to combine pieces of software that had been written by different parties in a way that would allow their runtime behaviors to be predictable with some measure of certifiability. The SEI's "predictable assembly from certifiable components" (PACC) work [Wallnau 2003] grew from the growing understanding of how the underlying architecture was essential to the effective use of COTS products. (See also the Architecture section.)

### 6.6.3 The Consequence: Effective Use of COTS Products

There has been a great deal of evolution in the last 15 years regarding the use of COTS products. Conversations about data rights and business models in connection with engineering and acquisition of systems are now routine. Processes and techniques have matured and evolved. Books covering the area have been published [Meyers 2001, Hissam 2001]. There was international participation in the International Conference on COTS-Based Software Systems (ICCBSS) for seven years, and papers delivered there are regularly cited in other publications. Rational/IBM saw the value in using EPIC as the basis for a plug-in for COTS package delivery [IBM 2010]. A supplement to Rational Method Composer 7.5, it is a guide for evaluating, recommending, acquiring, installing, configuring, fielding, and evolving COTS package solutions.

### 6.6.4 The SEI Contribution

When the SEI started investigating COTS-based systems, there was no other resource to which the federal government—most particularly, the DoD—could turn for guidance on achieving success with the use of COTS products. The SEI's influence extended beyond the Army, Navy, and Air Force, although courses were taught to all three. Other government organizations include the GAO, the Department of Justice, and the Department of Commerce.

The SEI developed a COTS-based systems body of knowledge based on broad experience. Through its CBS work, the SEI developed basic principles, techniques, and tools. SEI experts published guidance, presented tutorials at conferences, and provided training. It related CBS work to the CMMI, the Earned Value Method (EVM), Rational's RUP, and the spiral development method defined by Boehm. It used the International Standards Organization standard for software product evaluations [ISO/IEC 1998] as a starting place for developing the PECA process. The SEI also partnered with two other organizations to plan and lead the International Conference on COTS-Based Software Systems, held 2002-2008 (later renamed the International Conference on Component-Based Software Systems). The SEI's partners were the National Research Council, Canada (Ottawa, Canada) and the European Software Institute (Bilbao, Spain). This conference series attracted CBS researchers and practitioners from around the world and yielded a significant contribution to the CBS literature.

The value of this SEI work to the community is illustrated by the decision of Rational (now IBM) to use EPIC as the basis for a COTS plug-in for its tool suite called the RUP Plug-In for COTS Package Delivery [IBM 2010]. Also, Mälardalen University in Sweden held an annual workshop on component-based software engineering, and the SEI was involved in active working groups that developed from those workshops. The university created an ongoing research group based on interactions of the SEI with Professor Ivica Crnkovic, author of *Building Reliable Component-Based Software Systems* [Crnkovic 2002]. Finally, SEI work continues to be cited in journals and papers; see, for example, articles by Kusomo and by VanLeer [Kusomo 2012, VanLeer 2013].

The COTS-based systems work is an excellent example of the SEI's method of applied research, taking what is known about a subject, improving and enhancing it through research, and then packaging that in forms that meet the needs of government and industry customers.

### 6.6.5 References

[Albert 2002a] Albert, Cecilia & Brownsword, Lisa. *Evolutionary Process for Integrating COTS-Based Systems (EPIC): An Overview* (CMU/SEI-2002-TR-009). Software Engineering Institute, Carnegie Mellon University, 2002. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=6093

[Albert 2002b] Albert, Cecilia; Brownsword, Lisa; Bentley, David; Bono, Thomas; Morris, Edwin; & Pruitt, Deborah. *Evolutionary Process for Integrating COTS-Based Systems (EPIC) Building, Fielding, and Supporting Commercial-off-the-Shelf (COTS) Based Solutions* (CMU/SEI-2002-TR-005). Software Engineering Institute, Carnegie Mellon University, 2002. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=6053

[Boehm 1986] Boehm B. "A Spiral Model of Software Development and Enhancement." *ACM SIGSOFT Software Engineering Notes 11*, 4 (August 1986):14-24.

[Brownsword 1998] Brownsword, L.; Carney, D.; & Oberndorf, P. "The Opportunities and Complexities of Applying Commercial-Off-the-Shelf Components." *CrossTalk: The Journal of Defense Software Engineering 11,* 4 (April 1998): 4-6.

[Brownsword 2000] Brownsword, Lisa; Sledge, Carol; & Oberndorf, Patricia. *An Activity Framework for COTS-Based Systems* (CMU/SEI-2000-TR-010). Software Engineering Institute, Carnegie Mellon University, 2000. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=5225

[Carney 2003] Carney, David; Morris, Edwin; & Place, Patrick. *Identifying Commercial Off-the-Shelf (COTS) Product Risks: The COTS Usage Risk Evaluation* (CMU/SEI-2003-TR-023). Software Engineering Institute, Carnegie Mellon University, 2003. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=6767

[Comella-Dorda 2004] Comella-Dorda, Santiago; Dean, John; Lewis, Grace; Morris, Edwin; & Oberndorf, Patricia. *A Process for COTS Software Product Evaluation* (CMU/SEI-2003-TR-017). Software Engineering Institute, Carnegie Mellon University, 2004. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=6701

[Crnkovic 2002] Crnkovic, Ivica & Larsson, Magnus. *Building Reliable Component-Based Software Systems*. Artech House, 2002 (ISBN 1580533272).

[DSB 1994] Defense Science Board Task Force (Larry Druffel & George H. Heilmeier, co-chairs). *Acquiring Defense Software Commercially.* (Report #859), June 1994. http://www.dod.mil/pubs/foi/logistics_material_readiness/acq_bud_fin/859.pdf

[Hissam 2001] Hissam, Scott; Seacord, Robert C.; & Wallnau, Kurt C. *Building Systems from Commercial Products.* Addison-Wesley, 2001 (ISBN-10: 0-201-70064-6, ISBN-13: 978-0-201-70064-0).

[IBM 2010] IBM. *Rational Unified Process® (RUP®) Plug-ins for Rational Method Composer 7.5.* IBM 2010. http://www-01.ibm.com/support/docview.wss?uid=swg24028579.

[ISO/IEC 1998] ISO/IEC. *Software Product Evaluation, Part 5: Process for Evaluators* (ISO/IEC 14598-5:1998). ISO/IEC, 1998. http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=24906

[Kusumo 2012] Kusumo, D. S.; Staples, M.; Zhu, L.; Zhang, H.; & Jeffery, R. "Risks of Off-the-Shelf-Based Software Acquisition and Development: A Systematic Study and a Survey." *IET Seminar Digest 2012,* 1: 233-242.

[Meyers 2001] Meyers, B. Craig & Oberndorf, Patricia. *Managing Software Acquisition: Open Systems and COTS Products.* Addison-Wesley, 2001 (ISBN 0-201-70454-4).

[Packard 1986] Packard Commission (David Packard, Chairman). *A Quest for Excellence; Final Report to the President by the President's Blue Ribbon Commission on Defense Management,* June 1986. http://web.amsaa.army.mil/Documents/Packard%20Commission%20Report.pdf

[Tyson 2003] Tyson, Barbara; Alberts, Christopher; & Brownsword, Lisa. *Interpreting Capability Maturity Mode Integration (CMMI) for COTS-Based Systems* (CMU/SEI-2003-TR-022). Software Engineering Institute, Carnegie Mellon University, 2003. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=6759

[VanLeer 2013] VanLeer, M. D. & Jain, R. "A Framework to Assess the Impact of Systems of Systems Integration Using Commercially Off the Shelf (COTS) Technology." *Journal of Systems of Systems Engineering 4*, 1 (2013): 23-43.

[Wallnau 2001] Wallnau, Kurt; Hissam, Scott; & Seacord, Robert. *Building Systems from Commercial Components*. Addison-Wesley, 2001 (ISBN 0-201-70064-6).

[Wallnau 2003] Wallnau, Kurt. *Volume III: A Technology for Predictable Assembly from Certifiable Components* (CMU/SEI-2003-TR-009). Software Engineering Institute, Carnegie Mellon University, 2003. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=6633

## 6.7 Assurance Cases

### 6.7.1 The Challenge: Confidence in the Behavior of Performance-Critical Systems

A concern surrounding DoD systems was the need to shorten the certification process for safety, system reliability, or security. Traditional software and systems engineering techniques, including conventional test and evaluation approaches, were unable to provide the justified confidence needed. Consequently, a methodology to augment testing and evaluation was needed. The DoD needed to identify which parts are most important and which are less important, thereby enabling a more economically justified allocation of resources to the important points and issues. When a system had to be recertified because of a change to the system, the DoD needed to determine what must be changed, saving costly rework.

### 6.7.2 A Solution: Assurance Cases

The SEI had long experience in areas related to the DoD certification needs. SEI work in rate monotonic analysis (RMA) and in Simplex led to a more general interest in performance-critical systems, which became an area of concentration for the SEI in the mid-1990s. The SEI's investigation of performance-critical systems broadened the focus beyond real time, which had been the focus of RMA. The Simplex architecture allowed for improvements in performance without sacrificing the need to assure safety. Simplex provided a highly reliable core with higher performance code around this core area. If the higher performance code began to misbehave in some way, the system would automatically revert to the core, thereby providing higher performance when possible along with the assurance of safety in all situations. SEI work in dependable systems upgrades [Gluch 1997, 1998] was intended to extend this basic Simplex idea to a formal analysis of requirements and an effort to guarantee particular quality attributes.

Around the same time, the SEI became interested in fault-tolerant computing as it applied to software. SEI staff members developed a conceptual framework [Heimerdinger 1992], organized and hosted a series of dependable-software technology exchanges [Weinstock1993], and worked with the National Institute of Standards and Technology (NIST) to establish a Center for High Integrity System Software Assurance. As the SEI investigated performance-critical systems, and as systems of systems became more prevalent, the difficulty of assuring the safety, security, or reliability of net-centric systems of systems became clear—because of their size, complexity, and continuing evolution, and because they can exhibit undesired and unanticipated emergent behavior (actions of a system as a whole that are not simple combinations of the actions of the individual constituents of the system).

In considering the DoD certification problem, SEI experts focused on assurance cases. An assurance case provides a means to structure the reasoning that engineers use implicitly to gain confidence that systems will work as expected. It also becomes a key element in the documentation of the system and provides a map to more detailed information.

The concept of an assurance case was derived from the safety case, a construct that had been used successfully in Europe for more than a decade to document safety for nuclear power plants, transportation systems, automotive systems, and avionics systems. Much like a legal case presented in a courtroom, an assurance case requires arguments linking evidence with claims of conformance to the requirements of interest. It includes (1) a claim embodying what we want to show (e.g., the

system is safe); (2) evidence supporting the claim (e.g., a hazard analysis)—evidence can take on many forms, including test results, formal analyses, simulation results, fault-tree analyses, hazard analyses, modeling, and inspections, and (3) an argument explaining how the evidence is linked to the claim.

It is important that an assurance case be reviewable, which means that having a single claim ("The system does what it's supposed to do") and a single complex argument that links myriad evidence to the claim are not appropriate. Instead of taking such a large step, the claim is typically broken into subclaims, each of which can potentially be broken into yet another level of subclaims until the step to the actual evidence that supports that subclaim is almost obvious.

SEI work in assurance cases was initially funded by the NASA High Dependability Computing Project, beginning in 2002. On that project, the SEI worked with researchers at Carnegie Mellon University to introduce advanced thinking into NASA for use with various space projects, including the Mars Lander and the NASA Mission Data System (MDS), which had an unusual architecture that raised concerns about reliability. The SEI adapted ideas presented in a PhD thesis by Kelly at the University of York [Kelly 1998].

### 6.7.3 The Consequence: Assurance Cases Used in Practice

Application of assurance cases for certification has not been fully implemented yet, but it is an idea that many are considering and discussing. For example, NASA developed, with SEI assistance, an assurance case practice. For its Constellation project, the SEI contributed to NASA's safety requirements document, which specified the use of an assurance case to demonstrate safety. Although the assurance case requirement did not survive final review by NASA and its contractors, and the Constellation project subsequently was cancelled, the idea of assurance cases was distributed within NASA, leading to some research projects on assurance cases that continue to this day.

In 2006 the SEI was approached by the U.S. Food and Drug Administration (FDA). As a result of a number of safety incidents with infusion pumps, the FDA wished to improve the thoroughness of its review process and to improve the engineering done by manufacturers to assure safety. SEI work on the use of assurance cases in the development of medical devices [Weinstock 2009] led directly to the FDA's issuing draft guidance to manufacturers recommending the use of assurance cases and providing guidance for their use. As a result, infusion pump manufacturers are beginning to make use of assurance cases. The FDA is the only official agency of the U.S. government that has formally mandated the use of assurance cases to date.

### 6.7.4 The SEI Contribution

The idea that a structured argument is better than an unstructured argument is prevalent in Great Britain, where the Ministry of Defense (MoD) has for a decade or more required that an assurance-case kind of structure be presented for certain types of MoD systems. Subsequent to the start of the SEI's work in this area, the importance of assurance case concepts was recognized by the National Research Council in its report "Software for Dependable Systems: Sufficient Evidence?" [Jackson 2007].

The SEI has been instrumental in developing the assurance case from the existing European safety case concept, and showing how the cases can be used in various areas, such as aerospace and

medical devices. It is helping to create what could eventually become a routine practice of using assurance cases throughout the lifecycle to provide justified confidence that a system will perform as intended. Additionally, the SEI is developing a theory of argumentation that shows promise in helping to understand the contribution of specific evidence to system claims. The SEI has also applied assurance cases to claims about security and co-organized several workshops on that subject.

Work on assurance cases continues and is focused on creating a theory of argumentation that can be used to reason about the amount of confidence in a claim that is provided by particular pieces of evidence. The expectation is that this will lead to the ability to determine how to more effectively use scarce assurance resources. The theory is borrowing and extending concepts from law, philosophy, artificial intelligence, and other relevant disciplines.

### 6.7.5 References

[Gluch 1997] Gluch, David & Weinstock, Charles. *Workshop on the State of the Practice in Dependably Upgrading Critical Systems* (CMU/SEI-97-SR-014). Software Engineering Institute, Carnegie Mellon University, 1997. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=12775

[Gluch 1998] Gluch, David & Weinstock, Charles. *Model-Based Verification: A Technology for Dependable System Upgrade* (CMU/SEI-98-TR-009). Software Engineering Institute, Carnegie Mellon University, 1998. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=13105

[Heimerdinger 1992] Heimerdinger, Walter & Weinstock, Charles. *A Conceptual Framework for System Fault Tolerance* (CMU/SEI-92-TR-033). Software Engineering Institute, Carnegie Mellon University, 1992. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=11747

[Jackson 2007] Jackson, Daniel; Thomas, Martyns; & Millet, Lynn. *Software for Dependable Systems: Sufficient Evidence?* The National Academies Press, 2007 (ISBN 0309103940).

[Kelly 1998] Kelly, Timothy P. *Arguing Safety–A Systematic Approach to Managing Safety Cases.* PhD thesis. University of York, 1998. http://www.sei.cmu.edu/dependability/tools/assurancecase/upload/ArguingSafetyCases.pdf

[Weinstock 1993] Weinstock, Charles & Schneider, Fred. *Dependable Software Technology Exchange* (CMU/SEI-93-SR-004). Software Engineering Institute, Carnegie Mellon University, 1993. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=11785

[Weinstock 2009] Weinstock, Charles & Goodenough, John. *Towards an Assurance Case Practice for Medical Devices* (CMU/SEI-2009-TN-018). Software Engineering Institute, Carnegie Mellon University, 2009. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=8999

# 7 Architecture

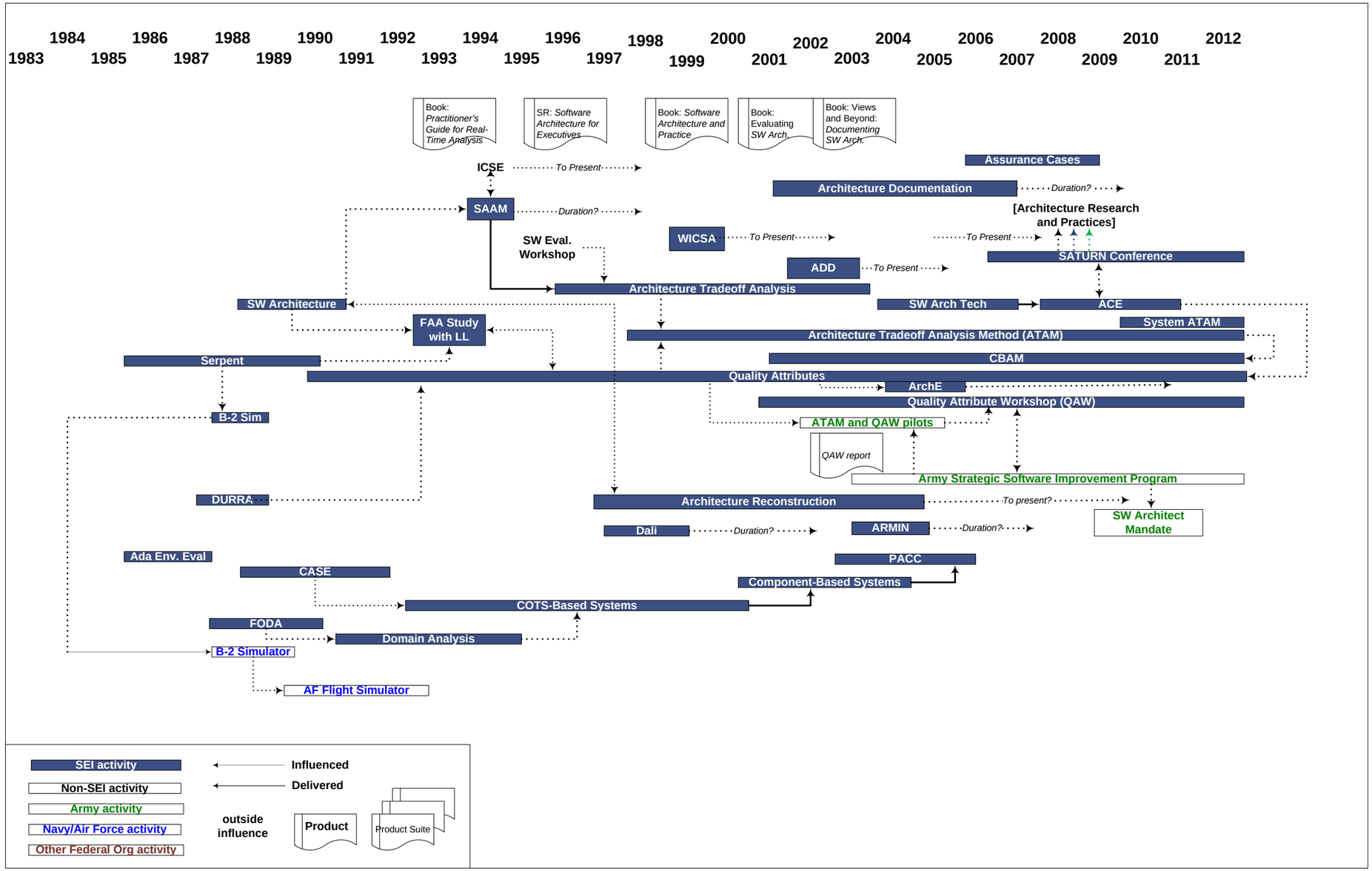


*Figure 7: Architecture Timeline*

## 7.0 Introduction to Software Architecture

At the time the SEI was established, there was little discussion of software architecture. Although there was a general recognition that the structure of software was important, that structure was often not visible or even documented. Indeed, structure could be many things—it was often an artifact of the methodology used for decomposition, or sometimes even a consequence of the way the system was decomposed for management across organizations, or the way the system modules were collected for execution on different processors.

This important asset was typically not controlled or maintained. Even when the structure was explicit, that structure was often violated inadvertently during evolution and almost always during maintenance. This was particularly true of real-time systems because they were mostly written in assembly language and the structure could be easily violated unintentionally.

Indeed, for most applications, there was little analysis of the best structure, except perhaps for automatic data processing (ADP) systems. Because of their large-database orientation, ADP systems began to evolve similar structures and methodologies appropriate to those applications that encouraged such consistency [Freeman 1982]. A few other applications were studied in great detail, most notably compilers. By the early 1980s, the structure of a compiler was so highly refined that researchers began developing tools to generate compilers [Johnson 1978]. But even in that highly developed area, the term *architecture* was seldom, if ever, used. Other well-studied applications that incorporated architectural principles, while not necessarily using the term, included relational database systems and operating systems.

Although there was little discussion of software architecture, there was significant discussion in the software engineering community on reuse of software components. There was also a growing recognition that effective reuse required consistent decisions or assumptions about the structure into which the components would be integrated. There were two DARPA programs addressing various aspects of software architecture with which the SEI collaborated, namely Software Technology for Adaptable, Reliable Systems (STARS) and Domain Specific Software Architectures (DSSA).

### 7.0.1 Seemingly Independent Efforts Prepared the SEI for an Early Consideration of Software Architecture

Several independent SEI efforts in the mid to late 1980s prepared the SEI for an early entry into a focus on software architecture. These included a structured modeling approach to building aircraft simulators, evaluation of the in-flight software system under development by the Federal Aviation Administration, and development of a generalized user interface management system. In parallel, there was a growing body of work elsewhere that influenced SEI thinking, including the DARPA STARS program, the Air Force Systems Command/Electronic Systems Division (AFSC/ESC) PRISM effort to define an architecture for command centers, and some commercial efforts.

The SEI was asked by the Air Force to evaluate perceived difficulties involving the application of Ada to the B2 simulator. Working with the contractor, the SEI team realized that the problem was not an Ada problem but that use of the more structured language made clear that the formerly acceptable approach violated modern software engineering principles. The SEI and the contractor evolved an approach modeled after the physical structure being simulated (the aircraft in this

case). This structural modeling approach enabled developers to reduce complexity and facilitate changeability. The SEI also developed a reference model and supporting tools. The benefits included significant reductions in test problems (from 2,000–3,000 to 600–700), in staff requirements for installation, in test expense, in defects, and in side effects from software changes.

The U.S. Department of Transportation and the DoD asked the SEI to join Lincoln Laboratory in conducting an independent technical assessment (ITA) of a problematic FAA upgrade of the in-flight air traffic control system. This FAA study included three FAA members contributing insights and documents. The team began to investigate the underlying software structure and concluded that it was sound and the upgrade project could be salvaged. The team provided a 14-point list of recommendations, which the FAA implemented. As a result, the contractor completed the upgrade, which went smoothly [Brown 1995]. The system is in use today. This experience encouraged the SEI to begin thinking more deeply about software architecture.

Earlier, the SEI had embarked on the development of a user interface management system (UIMS) intended to simplify the creation and modification of the user interface for interactive applications. Because the user interface is one of the most highly modified portions of most systems, special attention was needed to support the modification of the user interface. The Serpent UIMS consisted of a language for specifying the user interface, a compiler for that language, and a runtime engine to support the execution of the language. It was based on the architectural principle of separating the user interface from the remainder of the system to allow for the user interface to change without affecting any of the other code in the system. Again the notion of software architecture had become evident.

Because Serpent was one of a number of competing UIMSs, the SEI developed a method for comparing alternative designs for UIMSs to achieve the same functionality. This method—the Software Architecture Analysis Method (SAAM)—was useful in a more general context than the user interface domain and was based on recognizing that different systems, even in the same domain, may be created with different business goals. Business goals, such as length of use of the system, time to market, and execution within particular environments, led to different design choices. SAAM included a step in which these goals were explicitly stated so that design decisions could be evaluated against the goals.

The SEI first presented the SAAM at the International Conference on Software Engineering (ICSE) in 1994. The SAAM was based on the development and evaluation of scenarios for determining the ability to achieve defined business goals. Other methods in use at that time employed checklists (AT&T) and surveys (Siemens). The SAAM was seminal in the use of scenarios to perform architecture evaluation. Another scenario-based method, 4+1 View [Krutchen 1995], was developed concurrently at Rational. The SEI published a report for executives about software architecture in 1996 [Clements 1996].

SAAM led directly to the Architecture Tradeoff Analysis Method (ATAM), which evaluated a system for a collection of quality attributes (non-functional requirements) in addition to the modifiability of the user interface. Quality attribute requirements, such as those for performance, security, reliability, and usability, have a significant influence on the success of a system. The notion of paying attention to quality attributes first emerged from SEI work on Durra, a task-level appli-

cation description language, in the late 1980s. A focus on quality attributes in the context of business goals has been the consistent theme of the SEI's subsequent contributions to the field of software architecture.

The ATAM, which is in use today, is a method for evaluating software architectures relative to quality attribute goals. Since its development, the ATAM has emerged as the leading method in the area of software architecture evaluation. ATAM evaluations expose architectural risks that potentially inhibit the achievement of an organization's business goals. The ATAM got its name because it not only reveals how well an architecture satisfies particular quality goals, but it also provides insight into how those quality goals interact and trade off with each other.

### 7.0.2 Emergence of Architecture as a Separate and Well-Defined Area

By 1996, software architecture was emerging as a separate and increasingly well-defined area of interest within the field of software engineering. Two faculty members at Carnegie Mellon University published the first academic book on software architecture [Shaw 1996]. The SEI followed with a practitioner-oriented book [Bass 1997], and Siemens also published a book on software architecture during this time [Hofmeister 1999]. As interest in software architecture grew, the SEI started the Working IEEE/IFIP (Institute of Electrical and Electronics Engineers/International Federation for Information Processing) Conference on Software Architecture in San Antonio in 1999, providing a forum for the sharing of ideas and practices in the emerging field.

As the importance of software architecture became increasingly evident, the SEI began, in the late 1990s, to dedicate greater attention and resources to architecture-centric engineering. Results of this strategic focus include the following:

- **Architecture reconstruction, 1997**. This activity acknowledged the value of discovering implicit architectures through an examination of available evidence, with support from the SEI-developed Dali architecture-reconstruction tool and later the ARMIN (Architecture Reconstruction and MINing) tool, 2003. The SEI first applied architecture reconstruction in an engagement with the National Reconnaissance Office in 1998 [Kazman 1997, 2001].
- **Quality Attribute Workshop (QAW), 2001**. The QAW identifies important quality attributes and clarifies system requirements before there is a software architecture to evaluate. The SEI first began developing the concept of a QAW in work with the Deepwater Project for the Coast Guard in 1993. The QAW was derived from the ATAM. However, in its original incarnation, the QAW was tightly entwined with government acquisition and acquisition cycles. Understanding the utility of the technical core of the QAW, the SEI eventually developed a context-free instrument and began to apply it. It was codified in 2003 [Barbacci 2003], and a supporting toolkit was released in 2006.
- **Cost-Benefit Analysis Method (CBAM), 2001**. Another offshoot of the ATAM, the CBAM is an architecture-centric method for analyzing the costs, benefits, and schedule implications of architectural decisions [Kazman 2002].
- **Views and Beyond approach for documenting a software architecture, 2001.** In 2002, the first edition of *Documenting Software Architectures*, an influential and frequently cited book on the topic, was published in the SEI Series in Software Engineering by Addison-Wesley [Clements 2002].

- **ArchE (Architecture Expert), 2003.** This is a tool for moving from a set of quality attribute scenarios to an architecture design that satisfies those scenarios [Bachmann 2003].
- **Attribute-Driven Design (ADD) Method, 2006.** This is a method for designing the architecture of a software-intensive system by basing the design process on the architecture's quality attribute requirements [Wojcik 2006].
- **SEI Architecture Technology User Network (SATURN) Workshops, 2006**. The initial workshops, held in Pittsburgh, later evolved into the SATURN Conference series, an international forum for software architecture practitioners.
- **System ATAM, 2007.** This variant of the ATAM addresses system architecture notions and specifications, engineering considerations, quality attribute concerns, and architectural approaches.

### 7.0.3 Introduction of the Notion of Software Product Lines and Associated Practices

There had long been the expectation in the software community that software could be reused, and research and industry teams made various attempts to develop the mechanics for such a strategy, with varying levels of success. Predominantly, strategic reuse capitalizes on commonality—common features—and manages variation.

By the 1980s, diverse areas such as automobiles, aircraft, machine tools and, more recently, computer hardware, were using the concept of a product line, but the idea of a software product line was not common practice. The SEI was influenced to begin its formal investigation into software product lines by a number of related SEI experiences and by DoD and commercial attempts to create software product lines in the late 1980s and early 1990s. The reference architecture that the SEI developed for the B-2 simulator was later applied to other simulators, providing an early example of the potential for product lines. The SEI was also participating in the DARPA STARS program, which was experimenting with the development of software product lines, and in the AFSC/ESD PRISM project to experiment with the definition of a product line approach to the development of command centers. There were also a number of commercial industry efforts to define a software product line, most notably by CelsiusTech Systems AB.

The DARPA STARS program evolved from the effort that initially launched the SEI. The mission of STARS was to "[p]rovide DoD the technological, management and transitional basis to influence and enable a paradigm shift to a process-driven, domain-specific reuse based approach to software intensive systems" (from standard presentations about the program). Product lines and a development lifecycle that focused on commonality and variability were part of the global objective of STARS. As part of the STARS program, domain-analysis techniques were defined, architecturally oriented reuse library tools were developed, and three demonstration projects were initiated with the military services to pilot the tools and techniques. Successful application of tools and techniques in the Army and Air Force demonstration projects validated the efficacy of a product line approach but identified challenges to widespread adoption.

In 1991, the effort at AFSC/ESD called PRISM was motivated by two surveys that determined that 67 percent of Air Force-fielded command centers had functionally equivalent characteristics, while 75 percent of those fielded command centers had similar operational requirements.

AFSC/ESD conducted a source selection to find contractors to develop a model generic command-center architecture and functional specifications with supporting tools. ESD awarded research-and-development contracts to Raytheon and Hughes to build, test, and validate a "product line" approach to systems development [Hughes 1991]. TRW was later added as a third contractor. Eight line programs used the PRISM model architecture and supporting tools to realize an estimated 56 percent average savings in cost and an average 66 percent savings in time. These eight systems were early examples of software product lines.

At the time, the SEI was engaged in developing the Feature-Oriented Domain Analysis (FODA) [Kang 1990] that analyzes a problem domain across multiple similar systems to identify common and variable features. FODA serves as the basis for a vast number of subsequent feature modeling approaches and dialects still in use today. At the SEI, FODA later evolved into product line analysis, which extended the analysis of commonality and variability beyond features to quality attributes. The SEI investigation into product lines was also made possible by its concurrent focus on software architecture. SEI contributions in architecture definition, documentation, and evaluation were an important part of a software product line approach. Serendipitously, SEI staff members traveled to Sweden to interview staff at CelsiusTech Systems AB, ostensibly to do an architecture case study; what they found was that CelsiusTech had taken a product line approach that was achieving significant results in ship systems built for national defense, a domain of interest to the DoD. Those results included systems completed in days instead of years, order-of-magnitude productivity gains, and mass customization where 20 software builds were parlayed into a family of over a thousand specifically tailored systems. The promise of product lines that was documented in the CelsiusTech case study [Brownsword 1996] led the SEI to pursue an initiative in software product lines. The SEI recognized that when developing multiple similar products, there will be some degree to which they are the same, but there will also be some degree to which they vary. Economic advantage is achieved through a systematic product line approach that effectively manages this variation. Creating a software product line depends on establishing a software architecture, or product line architecture, for the entire set of systems.

### 7.0.4 Broad Use of SEI Approaches to Software Architecture

The influence of SEI work in software architecture on the DoD has been broad and pervasive. Major defense contractors, such as Boeing and Raytheon, now have architecture evaluation teams and architecture evaluation as part of their architect certification processes. Also, U.S. Army staff have reported repeatedly that use of scenario-based architecture evaluation methods reduces risk in schedule and cost, improves documentation, and results in higher-quality products.

Moreover, in 2009, the U.S. Army mandated that all Project Executive Offices appoint a chief software architect (CSWA) to be responsible for oversight and management of software development within each PEO. The memo specified that the CSWA must earn a Software Architecture Professional Certificate from the SEI (or another certificate-granting organization with software architecture expertise). The decision was based on an understanding of SEI work in software architecture and its impact and, in particular, a recent impact study of the use of SEI architecture evaluation techniques in the Army [SEI 2009]. Also, the SEI conducted a study of the impact of the Army Strategic Software Improvement Program (ASSIP)-sponsored QAWs and architecture evaluations using the ATAM. Ten out of 11 programs that responded to the survey indicated that ATAM/QAW produced better results than they traditionally obtained.

A measure of the influence of SEI architecture methods and techniques is their incorporation into a handbook for practitioners on the practice of architecting, written by an Open Group Master Certified IT architect. The book, T*he Process of Software Architecting* [Eeles 2010], advises practitioners on the importance of quality attributes and refers the reader to SEI techniques and methods (including quality attribute scenarios, architectural tactics, ADD, and ATAM).

Finally, IBM researcher Olaf Zimmerman sent the following note to the SEI, lauding its influence on the field of software architecture:

> *As briefly discussed at IMPACT, one cannot overestimate the importance of SEI assets have had, and continue to have, for numerous practicing architects in companies around the globe. This does not always become evident and tangible in public because most of these people spend all of their time on their projects, shipping products and satisfying clients; only few of them find the time to come to conferences or to become published authors. I can confirm this from my 10+ years in architectural roles in IBM professional services and development.*
>
> *The first two examples that come to my mind are:*
>
> - *When I received my company-internal IT architectural thinking and method training in the late 1990s, SEI books like "Software Architecture in Practice" were high up on recommended reading lists compiled by the creators of our methods; concepts like quality attributes of course play key roles in these methods.*
> - *Several of the clients in the telecommunications and finance sectors I worked with over the years look at the SEI not only for process advice and architectural education, but view the SEI as a trusted, independent advisor that regularly publishes high-quality reports on emerging concepts and technologies.*

As previously demonstrated in other areas, the SEI does not need to be the first to enter an area of investigation to be a leader in that area. The SEI contributions to architecture have been recognized internationally. This sentiment is best characterized by Philippe Kruchten in a keynote address at the Fourth Annual SEI SATURN Workshop held April 30–May 1, 2008 in Pittsburgh. He said,

> *Over the last 15 years, the SEI has become a sort of mecca for software architecture—a place where anyone who is doing any work related to software architecture must go.*

### 7.0.5 References

[Bachmann 2003] Bachmann, Felix; Bass, Len; & Klein, Mark. *Preliminary Design of ArchE: A Software Architecture Design Assistant* (CMU/SEI-2003-TR-021). Software Engineering Institute, Carnegie Mellon University, 2003. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=6751

[Barbacci 2003] Barbacci, Mario; Ellison, Robert; Lattanze, Anthony; Stafford, Judith; Weinstock, Charles; & Wood, William. *Quality Attribute Workshops (QAWs)*, 3rd ed. (CMU/SEI-2003-TR-016). Software Engineering Institute, Carnegie Mellon University, 2003. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=6687

[Bass 1997] Bass, Len; Clements, Paul; & Kazman, Rick. *Software Architecture in Practice*, 1st ed. Addison-Wesley Professional, 1997 (ISBN 0201199300). (3rd ed. published 2012, ISBN 0321815734).

[Brown 1995] Brown, Alan; Carney, David J.; Clements, Paul C.; Meyers, B. Craig; Smith, Dennis B.; Weiderman, Nelson H.; & Wood, William G. "Assessing the Quality of Large, Software-Intensive Systems: A Case Study." *Proceedings of the 5th European Software Engineering Conference (ESEC '95)* (Published as Lecture Notes in Computer Science 989) Sitges, Spain, September 25-28, 1995. Springer, 1995.

[Brownsword 1996] Brownsword, Lisa & Clements, Paul. *A Case Study in Successful Product Line Development* (CMU/SEI-96-TR-016). Software Engineering Institute, Carnegie Mellon University, 1996. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=12587

[Clements 1996] Clements, Paul & Northrop, Linda. *Software Architecture: An Executive Overview* (CMU/SEI-96-TR-003). Software Engineering Institute, Carnegie Mellon University, 1996. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=12509

[Clements 2002] Clements, Paul; Bachmann, Felix; Bass, Len; Garlan, David; Ivers, James; Little, Reed; Nord, Robert; & Stafford, Judith. *Documenting Software Architectures: Views and Beyond*, 1st ed. Addison-Wesley Professional, 2002 (ISBN 201703726). (2nd ed. published 2010, ISBN 0321552687).

[Eeles 2010] Eeles, Peter & Cripps, Peter. *The Process of Software Architecting*. Addison-Wesley Professional, 2010 (ISBN 0321357485).

[Freeman 1982] Freeman, Peter & Wasserman, Anthony. *Software Methodologies and Ada*. National Technical Information Service, 1982.

[Hughes 1991] Hughes, David. "ESD Changes Procurement Strategy for Software to Drive Command Centers; Air Force Electronic System Division's PRISM Program." *Aviation Week & Space Technology 135* (August 26, 1991): 56-57.

[Johnson 1978] Johnson, Stephen C. "Yet Another Compiler." *Bell Labs Computer Science Technical Report 32*. Bell Labs, 1978.

[Hofmeister 1999] Hofmeister, Christine; Nord, Robert; & Soni, Dilip. *Applied Software Architecture.* Addison-Wesley Professional, 1999 (ISBN 0321643348).

[Kang 1990] Kang, Kyo; Cohen, Sholom; Hess, James; Novak, William; & Peterson, A. *Feature-Oriented Domain Analysis (FODA) Feasibility Study* (CMU/SEI-90-TR-021). Software Engineering Institute, Carnegie Mellon University, 1990. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=11231

[Kazman 1997] Kazman, Rick & Carriere, S. *Playing Detective: Reconstructing Software Architecture from Available Evidence* (CMU/SEI-97-TR-010). Software Engineering Institute, Carnegie Mellon University, 1997. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=12887

[Kazman 2001] Kazman, Rick; O'Brien, Liam; & Verhoef, Chris. *Architecture Reconstruction Guidelines* (CMU/SEI-2001-TR-026). Software Engineering Institute, Carnegie Mellon University, 2001. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=5741

[Kazman 2002] Kazman, Rick; Asundi, Jayatirtha; & Klein, Mark. *Making Architecture Design Decisions: An Economic Approach* (CMU/SEI-2002-TR-035). Software Engineering Institute,

Carnegie Mellon University, 2002. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=6265

[Kruchten 1995] Kruchten, Philippe. “Architectural Blueprints: The ‘4+1’ View Model of Software Architecture.” *IEEE Software 12,* 6 (November 1995): 42-50.

[SEI 2009] Software Engineering Institute. “Army Requires PEOs to Appoint Chief Software Architect.” *2009 Year in Review.* Software Engineering Institute, Carnegie Mellon University, 2010.

[Shaw 1996] Shaw, Mary & Garlan, David. *Software Architecture: Perspectives on an Emerging Discipline*. Prentice-Hall, 1996 (ISBN 0131829572).

[Wojcik 2006] Wojcik, Rob; Bachmann, Felix; Bass, Len; Clements, Paul; Merson, Paulo; Nord, Robert; & Wood, William. *Attribute-Driven Design (ADD), Version 2.0* (CMU/SEI-2006-TR-023). Software Engineering Institute, Carnegie Mellon University, 2006. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=8147

## 7.1 Structural Modeling

### 7.1.1 The Challenge: Efficiently Replicating Aircrew Trainers

The U.S. Air Force has a long history of using aircrew trainers as an integral part of aircrew training programs. These trainers are expensive to build. However, the benefits of trainers justify the expense. Aircrew trainers reduce training costs, improve safety, support security, and provide flexibility and convenience [Rolfe 1988].

Flight simulators train pilots for flight missions in a safe, convenient, and cost-efficient way. To effectively provide this training, the software for a flight simulator must work with the hardware to faithfully reproduce the behavior of the aircraft being simulated. Flight simulator software, therefore, must perform in real time, be developed and continually altered to keep pace with the technological advances in the simulated aircraft, and undergo a validation process that certifies acceptability as a pilot training device. In addition to this inherent complexity, the flight simulator software is typically very large in scale, in the range of one million lines of high-level language code. For example, the B-2 trainer contained more than 1.7 million lines of Ada simulation code.

The scale and complexity of the software precipitated numerous problems in both the developed product and the development and evolution processes. Modifications to the simulator software severely lagged behind modifications to the aircraft being simulated, resulting in a software product that did not faithfully simulate the current aircraft. As is often the case in large-scale software development efforts, geographically remote software teams were concurrently developing parts of the flight simulator system. The time required to integrate these parts, developed by the diverse work teams, was growing at alarming rates. The correction of errors in the simulator software was complicated and time consuming. Modifications to add functionality were also time sinks. Often the cost of modifications to the software exceeded the software development cost.

Moreover, certain flight behaviors were becoming increasingly difficult to simulate because the organization of functionality in the simulator was different from that in the physical aircraft. The unwieldy software architecture reduced effective communication of the design in reviews and interactions among the work team members. Ultimately, this lack of effective communication decreased the control and visibility of the software to the point where it created risk in development and maintenance.

### 7.1.2 A Solution: Structural Modeling

The recognition of the technical risk with simulators was the catalyst that drove the development of the structural modeling method. Structural modeling is an object-based software engineering strategy developed by the collaborative efforts of the U.S. Air Force, Air Force contractors, and the SEI. The broad objective behind structural modeling was to take a problem domain of great complexity and scale and to abstract it to a coarse enough level to make it manageable, modifiable, and able to be communicated to a diverse user and developer community.

Work on structural modeling began in 1986, when Air Force engineers recognized that the traditional software architecture for flight simulators was reaching its limits. The SEI initially responded to a request by the Air Force by initiating an effort to assist with the simulator for the B-2 (at that time a classified program). What the contractor and the Air Force at first thought was a problem in applying Ada, the SEI quickly realized was a conceptual problem in applying notions

of abstraction to a FORTRAN-oriented structure that was typical for many simulator software developers. After the initial problem was resolved, the Air Force and the SEI recognized an opportunity to focus on the structure of software for simulators, which the SEI dubbed "structural modeling." The SEI developed a reference architecture for aircraft simulators with supporting tools to enable simulator developers to develop simulators more rapidly and economically [Howard 1993].

The initial goal of the joint effort was to reduce the complexity of producing aircrew trainer simulation software, particularly complexity as encountered during software integration. Structural modeling was based on the realization that the difficulties and concerns with the evolution of aircraft and simulators represented design issues to be addressed by the software architecture. It can be understood as a design effort to produce a software architecture that supports changeability. The general software design principles that were applied included separation of domain commonality from variability (separating the code expected to remain constant for a domain from the code expected to change); object-based partitioning based on the physical structure of the simulated aircraft; separation of concerns; restriction of interfaces and mechanisms for communication between components; and restriction of the flow of control [Abowd 1993, Chastek 1996].

### 7.1.3 The Consequence: Efficient Reuse of a Reference Model for Aircrew Trainers

In a series of interviews, experts from organizations using structural modeling reported improvements that included significant reductions in the following:

- *Test problems.* Test descriptions (test problems) were reduced from 2,000–3,000 in a previous data-driven simulator of comparable size (the B-52) to 600–700 test descriptions with structure modeling.
- *Staff requirements for installation.* The need for onsite staff during initial installation and use was reduced by about 50 percent. Fault detection and correction were significantly easier.
- *Test expense*. Staff typically could isolate a reported problem offline rather than going to a site. Reduced testing and fault isolation on the actual trainer was particularly significant given the high expense of trainer time.
- *Side effects from software changes.* Most projects noted that side effects from software changes were a rare occurrence, primarily due to the encapsulation of functionality in subsystems.
- *Defects.* With the use of structural modeling, defect rates for one project were half that found on previous data-driven simulators.

Another reported improvement was extreme ease of integration. All projects commented on the lack of the "big-bang" effect compared to their previous data-driven simulators of comparable size. The use of common structural types for components and a standard mechanism for communication between components were cited as key contributors. Also, there was significant improvement in reuse across trainers. One project reported reuse of the software architecture, executive, and subsystem controllers.

### 7.1.4 The SEI Contribution

As a result of the collaboration among the SEI, the Air Force, and Air Force contractors, a culture of structural modeling evolved. Structural modeling experience was gained in a number of simulator acquisitions. While data specific to those acquisitions are not generally available, there have been some internal reports within the SEI and Air Force that have described portions of the object-based technology underlying the structural modeling approach [Lee 1988, USAF 1993]. The following projects used structural modeling with SEI support:

- B-2 Weapons System Trainer, started in 1986, with 1.7 million lines of Ada code developed by Hughes Training/Link Division. Hughes/Link continued to maintain the system.
- C-17 Aircrew Training System (ATS), started in 1990 with 350,000 lines of Ada code developed by McDonnell Douglas. McDonnell Douglas owned the software and performed the training for the Air Force.
- Special Operations Forces (SOF) ATS supporting the C-130, started in 1991 with 750,000 lines of Ada code developed by Loral Federal Systems. One-half of the system used structural modeling.
- Simulator Electric Combat Training (SECT), started in 1992 with 250,000 lines of Ada code developed by AAI Corporation. Three-quarters of the system used structural modeling.

Once a reference architecture was established for the B-2 simulator that appealed to the aircraft simulator community, the SEI was encouraged to help the Air Force apply a similar approach to other aircraft simulators. The SEI soon realized that the same reference architecture could be incorporated and began to build tools to simplify the process of generating the structure as abstractions of the physical components of the aircraft. Reapplication of the reference architecture and growing set of tools enabled the simulator SOF to develop a family of simulators. This led the SEI to begin thinking about application of these notions to other applications, ultimately contributing to evolution of the SEI's seminal and influential work in software product lines.

### 7.1.5 References

[Abowd 1993] Abowd, Gregory; Bass, Len; Howard, Larry; & Northrop, Linda. *Structural Modeling: An Application Framework and Development Process for Flight Simulators* (CMU/SEI-93-TR-014). Software Engineering Institute, Carnegie Mellon University, 1993. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=11899

[Chastek 1996] Chastek, Gary & Brownsword, Lisa. *A Case Study in Structural Modeling* (CMU/SEI-96-TR-035). Software Engineering Institute, Carnegie Mellon University, 1996. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=12675

[Howard 1993] Howard, Larry & Bass, Len. "Structural Modeling for Flight Simulators," 876-881. *Proceedings of the 1993 Summer Computer Simulation Conference (SCSC '93)*. Boston, MA, July 19–21, 1993.

[Lee 1988] Lee, K.; Rissman, M.; D'Ippolito, R.; Plinta, S.; & Van Scoy, R. *An OOD Paradigm for Flight Simulators*, 2nd ed. (CMU/SEI-88-TR-30, ADA204849). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1989. http://resources.sei.cmu.edu/asset_files/TechnicalReport/1993_005_001_16190.pdf

[Rolfe 1988] Rolfe, J. M. & Staples, K. J. *Flight Simulation*. Cambridge University Press, 1988 (ISBN 0521357519).

[SEI 1992] Software Engineering Institute. "Reducing Technical Risk with Structural Modeling," *Bridge* (a quarterly publication of the SEI). December, 1993.

[USAF 1993] United States Air Force. *An Introduction to Structural Models* (USAF ASC-TR-935-5008). Wright-Patterson AFB, 1993.

## 7.2 Federal Aviation Administration Study

### 7.2.1 The Challenge: Evaluate a Problematic FAA System Under Development

By 1995, the FAA had spent $2 billion on a project to upgrade the in-flight air traffic control (ATC) system. The attempt to upgrade the system had gone through many well-documented and costly delays [Glass 1997]. At that time, it continued to fail some of its system tests; as a result, it still had not been put into service. The existing system, which had been launched a number of years earlier, had obsolete hardware and software and was becoming very difficult to maintain. In addition, because of rapidly expanding hardware advances, there were not enough replacement hardware parts available (mostly transistors), and the hardware in use was slow.

### 7.2.2 A Solution: SEI Architecture Evaluation Methods

The SEI was contacted by senior officials at the U.S. Department of Transportation (DoT) and the DoD to conduct an independent technical assessment to determine if the system could be salvaged. This work by the SEI, in a joint project with Lincoln Laboratory, represented one of the first times that the SEI had done work on behalf of a non-DoD federal agency. Lincoln Laboratory personnel had domain knowledge of ATC systems and knew the FAA well, and the joint SEI/Lincoln Laboratory team worked together very successfully. The FAA also provided the team with three members to contribute insights from the FAA perspective and provide unfettered access to personnel and documents. The team was asked to provide input to FAA management on the technical question of whether the upgraded system was salvageable and, if so, what would need to be done to feel confident that the system would perform at an acceptable level. If the system was not salvageable, the FAA was prepared to walk away from the two billion dollar investment that it had made.

The SEI/Lincoln Laboratory/FAA team split into subgroups to investigate the areas of modifiability, availability, scheduling, code quality, documentation, software tools, maintenance environment, testing, change management, and software processes.

To address the question of modifiability, the SEI developed a set of likely change scenarios. At that time, the SEI was just beginning to develop scenario-based methods for evaluating software architectures. The prime contractor was asked to demonstrate the amount of change that would be required for each of the likely changes to the system. As a result, the subgroup determined that with regard to modifiability, significant changes could be put in place with relatively little disruption to the system. This was the first practical application of a method for assessing the quality attribute of modifiability, and this method formed part of the foundation for the SEI Software Architecture Analysis Method. In turn, after much future analysis, some of the ideas for SAAM later evolved into the Architecture Tradeoff Analysis Method and the Quality Attribute Workshop.

The other subgroups focused on their respective topics. One group analyzed the quality attribute of availability because the system had been developed to have extensive hardware and software redundancy, and many of the decisions on the structure of the system had been made to ensure availability. The availability assumptions were examined; while many different structures for availability could have been developed, the one in use was judged to be adequate.

Another group inferred the quality of code by examining code samples. The group also analyzed samples of the documentation for accuracy, readability, and consistency. The other subteams examined the methods and tools that were to be used in both the development and maintenance environments, the change-management model, the use of schedulability models, and the processes that had been used. Several observations were that the FAA maintenance staff in Atlantic City was using obsolete technology with no relationship to the tools and methods that the contractor had adopted. In addition, the proposed requirements for interoperability of the tools were unprecedented and had not been demonstrated to work. There had not previously been a rigorous schedulability analysis even though this method was now common practice. The subteam also analyzed the change-management process. While the overall model was sound, the group made recommendations for setting more effective priorities on needed changes and tracking them.

The general conclusion of the assessment team was that the proposed new architecture was good, the code was fair, and the documentation was poor. The team provided 14 recommendations, both technical and non-technical, for the contractor and the FAA to follow if the decision was made to keep the existing system.

### 7.2.3 The Consequence: Successful FAA System Upgrade

The assessment team's 14 recommendations had significant impact because the inspector general for the DoT was following this assessment closely, as was the General Accounting Office[58] (GAO) [GAO 1999]. After the study was completed, the FAA was directed by the DoT and GAO to adhere to these 14 points. The recommendations were adopted within the FAA as well as by the contractor. As a result of the team's study, the FAA gave the contractor a fixed-price contract to complete the work. This contract incorporated each of the 14 points of the SEI- Lincoln Laboratory team. The contractor completed the work, the system was upgraded, and this system is in use today.

The upgrade process went smoothly enough that all parties involved were able to declare success.

Another positive result of the experience was that the FAA contracted with the SEI for a number of follow-up projects. This project also represented the first of many SEI independent technical assessments of non-DoD government acquisition projects. The team documented its approach, and it was used as a model for a number of these projects. In addition, the team wrote several articles describing the project to the broader research community and presented these papers at two technical conferences in 1995, the European Software Engineering and Foundations of Software Engineering conferences [Brown 1995a, 1995b].

### 7.2.4 The SEI Contribution

This entire effort was a collaboration between Lincoln Laboratory and the SEI. Lincoln Laboratory provided the domain knowledge as well as the operational understanding of the FAA. Without that expertise, the effort would have had a low probability of success. The SEI contributed its

58 At the time this was the General Accounting Office; it is now the Government Accountability Office.

evolving understanding of software architectures and how they could be documented and assessed. As a result of this effort, the SEI was able to demonstrate many of the underlying concepts and it served as a basis for further refinement in subsequent software architecture-related efforts.

### 7.2.5 References

[Brown 1995a] Brown, Alan; Carney, David J.; Clements, Paul C.; Meyers, B. Craig; Smith, Dennis B.; Weiderman, Nelson H.; & Wood, William G. "Assessing the Quality of Large, Software-Intensive Systems: A Case Study." *Proceedings of the Fifth European Software Engineering Conference (ESEC '95)* (Published as Lecture Notes in Computer Science 989) Sitges, Spain, September 1995. Springer, 1995.

[Brown 1995b] Brown, A. W.; Carney, D. J.; & Clements, P. "A Case Study in Assessing the Maintainability of Large, Software-Intensive Systems," 240-247. *Proceedings of the 1995 International Symposium on the Frontiers of Software Engineering.* Tucson, Arizona, 1995. IEEE Computer Society, 1995.

[GAO 1999] General Accounting Office. *Observations on FAA's Air Traffic Control Modernization Program*. Testimony Before the Subcommittee on Commerce, Science and Transportation, U.S. Senate, March 1999. http://www.gao.gov/archive/1999/r199137t.pdf

[Glass 1997] Glass, Robert, L. *Software Runaways*. Prentice Hall, 1997 (ISBN 013673443X).

## 7.3 Reducing the Cost of Modifying the User Interface

### 7.3.1 The Challenge: Cost-Effectively Modifying User Interfaces for Defense Systems

In studies dating back to 1978, data showed that the cost of development and modification of the user interface contributed over 50 percent to the total cost of ownership [Sutton 1978]. Attempts to reduce the cost of developing defense systems clearly had to include reduction in the cost of developing and maintaining the user interface.

### 7.3.2 A Solution: The Serpent User Interface Management System

The high cost of developing and modifying the user interface led to a class of systems intended to reduce this cost. These user interface management systems (UIMSs) varied in their target audience (developers and end users) and in their approaches. Serpent was created and distributed by the SEI to demonstrate UIMS concepts to developers of DoD software-intensive systems. In addition, SEI staff published articles about Serpent (for example those by Bass and Lee [Bass 1989, 1990; Lee 1990].

In the general case, a UIMS is a set of tools for the specification and execution of the user interface portion of the system. A UIMS provides tools for the specification of the static, layout portion and the dynamic portion of the user interface, and for the execution of the specifications.

Serpent was a UIMS that approached the problem of reducing the total ownership cost of the user interface by separating the user interface portion of a system from the functional portion, allowing for modifications to the user interface with minimal impact on the remainder of the system. The user interface is a major concern of most computing systems and, generally, is distinct from the concerns of the application. Separating the user interface from the application leads to a three-part division of a software system: the presentation of the user interface, the functionality of the application, and the mapping between the user interface and the application.

The advantages of this division are the following:

- It allows modifications of the user interface to be done with minimal modification to the functional portion and vice versa. It does this by isolating the functional portion of the application from the details of the user interface. For example, whether a command is specified through a menu choice or through a textual string is not relevant to the functional portion of an application. Removing these concerns from the functional portion of the application allows the type of interface to be modified dramatically without any modifications to the application.
- It allows the development of tools that are specialized for the design, specification, and execution of the user interface. For example, a layout editor, a dynamic specification language, and a runtime to support them can be included.

Serpent supported the incremental development of the user interface from prototyping through production and maintenance. It did this by providing an interactive layout editor for prototyping, by integrating the layout editor with a dynamic specification language for production and maintenance, and by having an open architecture so that new user interface functionality could be added during the maintenance phase.

The basic features of Serpent were simple enough for use during the prototyping phase, yet sophisticated enough for use in developing the prototype into an operational system. Serpent was designed to be extensible in the user interface toolkits that can be supported. Hence, a system developed using Serpent could be migrated to new technologies without time-consuming and expensive reengineering of the application portion. Serpent consisted of a language designed for the specification of user interfaces, language to define the interface between the application and Serpent, a transaction processing library, an interactive editor for the specification of dialogues and for the construction and previewing of displays, and a variety of input/output (I/O) technologies.

Serpent provided many features to address the requirements, development, and maintenance phases of a project. For the requirements phase, Serpent provided a language and an editor to define the user interface. For the development phase, Serpent provided a set of tools that simplify the development of the user interface. For the maintenance phase, Serpent allowed integration of new technologies as well as the ability to modify the user interface. Specifically, Serpent did the following:

- It provided generality in supporting a wide range of both applications and I/O toolkits through its use of database-like schemas.
- It provided a set of tools that simplified the user interface implementation process.
- It encouraged the separation of software systems into user interface and "core" application portions, a separation that would decrease the cost of subsequent modifications to the system.
- It supported rapid prototyping and incremental development of user interfaces.
- It facilitated the integration of new user interface toolkits into the user interface portion of a system.
- It supported both synchronous and asynchronous communication, allowing real-time applications to satisfy timing constraints without waiting for user input.

### 7.3.3 The Consequence: Understanding the Relationship Between the User Interface and Software Architecture

The prototypes developed during this line of worldwide research into UIMS were not used directly for the development of software. Rather, the effort sensitized a generation of user interface researchers to the impact of software engineering architectural decisions on the ease of modifying the user interface, and it contributed an important concept to the discipline of software architecture that emerged in the 1990s.

Current DoD systems, such as the Command Post of the Future, now follow this separation of user interface. DoD Program Offices, such as Force XXI Battle Command, Brigade-and-Below (FBCB2), often require such separation.

### 7.3.4 The SEI Contribution

Much of the work on the development of different software architecture models for user interface construction was done by the International Federation of Information Processing Working Group on User Interface Engineering (IFIP WG2.7/13.4) as well as by ad hoc groups of user interface software developers. Because of Serpent, the SEI was able to make significant contributions both in the work of IFIP WG2.7 as its members developed Presentation-Abstraction-Control (PAC)

and in organizing an ad hoc working group of user interface software developers that developed Arch/Slinky [Bass 1992].

Serpent was one of a number of user interface models introduced from 1980–1990. In addition to PAC and Arch/Slinky, these include the Model View Controller and the Seeheim Model. These models are still being covered in standard Human Computer Interaction textbooks [Dix 2003].

### 7.3.5 References


[Bass 1989] Bass, L. "Serpent." *ACM CHI '89 Human Factors in Computing Conference.* Austin, Texas, April 30-June 4, 1989. ACM, 1989.

[Bass 1990] Bass, L.; Clapper, B.; Hardy, E.; Kazman, R.; & Seacord, R. "Serpent: A User Interface Environment." *1990 USENIX Winter Conference*, Washington, D.C., January 1990.

[Bass 1992] Bass, L.; Faneuf, R.; Little, Reed; Mayer, Niels; et al. "A Metamodel for the Runtime Architecture of an Interactive System," UIMS Tool Developers Workshop. *SIGCHI Bulletin 24,* 1 (January 1992): 34-37.

[Dix 2003] Dix, Alan; Finlay, Janet E.; Abowd, Gregory D.; & Beale, Russell. *Human-Computer Interaction,* 3rd ed. Prentice-Hall, 2003 (ISBN 0130461091).

[Lee 1990] Lee, Ed; Hall, Frank; Bowers, Andrea, Yang, Sherry; Bass, Len; Lemke Andreas; & Shan, Yen-Ping. "User-Interface Development Tools." *IEEE Software 7*, 3 (May 1990): 31-36.

[Sutton 1978] Sutton, Jimmy A. & Sprague, Ralph H. Jr. *A Study of Display Generation and Management in Interactive Business Applications*. (Technical Report RJ2392), IBM Research, November 1978.

## 7.4 Software Architecture Analysis Method

### 7.4.1 The Challenge: Predicting Systems Development Problems in Advance

One of the recurring themes of the various defense challenge problems is that of predicting problems before a system has been built. Maintenance and improvement costs represent more than half the total cost of a system, and this percentage has been steadily growing since 1960 [Jones 2006]. The problem for DoD is to predict maintenance problems before the system is constructed and before these problems occur.

### 7.4.2 A Solution: The Software Architecture Analysis Method

The Software Architecture Analysis Method grew out of an effort undertaken by members of the Serpent user interface project after the UI project was officially ended. The members of the UI project were active participants in the Interface Federation of Information Processing Working Group on User Interface Engineering (IFIP WG 2.7/13.4). This group was composed of members interested in user interface management systems (UIMSs), and the group had many discussions that centered on the thesis "my system is better than your system." From these discussions, the SEI members of the working group gathered two principles that were instrumental in the SAAM and that have been embodied in almost every software architecture evaluation method since the SAAM:

- Different developers have different business goals for their systems. An architecture evaluation must evaluate an architecture against the business goals for the system. This means the architecture evaluation method must either explicitly state what goals it is evaluating or elicit the goals as part of the evaluation.
- Evaluating for a property such as maintainability without a specific definition of what *maintainability* means is pointless. Every system is modifiable for some set of modifications and not modifiable for others.

The importance of understanding the business goals prior to evaluating an architecture can be seen in the claims made for various UIMSs that led to the development of the SAAM. Examples of these claims are:

- "We have developed … user interface components that can be reconfigured with minimal effort." [Pittman 1990]
- "Serpent … encourages the separation of software systems into user interface and 'core' application portions, a separation which will decrease the cost of subsequent modifications to the system." [Bass 1989]
- "This Nephew UIMS/Application interface is better [than] traditional UIMS/Application interfaces from the modularity and code reusability point of views." [Sleekly 1989]

Each of these systems, which nominally are supposed to serve the same purpose, has different goals. Thus, comparing or evaluating architectures must be done in light of the system's goals. Whether these are the correct goals is outside of the scope of an architecture comparison or evaluation.

Even if two systems have the same goal—for example, making the system more modifiable—evaluating an architecture against that goal directly is not possible because the set of potential modifications is unbounded. Modifying an avionics system to do logistic planning is neither desirable nor feasible. Enabling the evaluation of an architecture for a particular property requires being specific about the types of modifications that are anticipated. The type of anticipated modifications also reflects the business goals for the system. Will a particular system subsequently be integrated into a system of systems? Is the replacement of a particular system in the planning process? The answers to these questions will determine what set of modifications are likely and how flexible the architecture for a system needs to be.

**The View from Others**

*SAAM is the first widely promulgated scenario-based software architecture analysis method.*

– Mugurel T. Ionita, Department of Mathematics and Computing Science, Technical University Eindhoven, The Netherlands [Ionita 2002]

The SAAM required that the business owner of a system specify anticipated changes through the use of "change scenarios." A change scenario specifies that "this system should be modified to perform this specific new additional function." The SAAM calls for the architect to describe what modifications will be made to the system to achieve the new functionality. The extent of these changes is then used to measure how difficult the changes will be to implement. The business owner of the system must then decide whether the anticipated difficulty is within bounds or whether the architecture design should be modified to make the anticipated changes easier.

The steps that the SAAM implemented were to develop scenarios, describe the architecture of the system under consideration, evaluate the difficulty of each individual scenario, assess scenario interactions, and create the overall evaluation. The consolidated final output of a SAAM evaluation was a rating of the architecture with respect to the total set of scenarios.

### 7.4.3 The Consequence: Robust Multi-Quality Architectural Evaluation Method

The SAAM was seminal in the use of scenarios to perform architecture evaluation. An immediate outcome of applying the SAAM in practice was that stakeholders did not want a method that only evaluated a system for a single quality attribute. The SAAM led directly to the Architecture Tradeoff Analysis Method, which evaluated a system for a collection of quality attributes.

Major defense contractors, such as Boeing and Raytheon, now have architecture evaluation teams and architecture evaluation as a portion of their architect certification process. The U.S. Army staff reported repeatedly that use of the scenario-based architecture evaluation methods reduced risk in schedule and cost, improved documentation, and resulted in a higher quality product [SEI 2009].

### 7.4.4 The SEI Contribution

The SEI pioneered the use of scenario-based methods in the evaluation of software architectures. The SAAM introduced the concept of a quality attribute scenario, giving specific modifications against which the system is to be tested. Later evaluation methods generalized these modifiability scenarios to quality attribute scenarios.

A survey of software architecture evaluation methods published in *IEEE Transactions on Software Engineering* identifies six different scenario-based evaluation methods, all of which cite SAAM as one of their sources [Dobrica 2002].

SEI staff members published three scholarly articles on the SAAM [Kazman 1994, 1996; Clements 1995], and they had been cited by others 441, 410, and 58 times, respectively as of August 2014. The SEI also participated in industry groups to document techniques used in software architecture evaluation [Kruchten 2001].

The SEI teaches courses and certifies individuals as competent in scenario-based architecture evaluation methods.[59]

### 7.4.5 References


[Bass 1989] Bass, L. *Serpent Overview* (CMU/SEI-89-UG-2). Software Engineering Institute, Carnegie Mellon University, 1989. resources.sei.cmu.edu/asset_files/WhitePaper/2007_019_001_29297.pdf

[Clements 1995] Clements, Paul; Bass, Len; Kazman, Rick; & Abowd, Gregory. "Predicting Software Quality by Architectural Evaluation," 485-497. *Proceedings of the Fifth International Conference on Software Quality,* Austin, Texas, October 1995.

[Dobrica 2002] Dobrica, Lillian; & Niemela, Elia. "A Survey on Software Architecture Analysis Methods." IEEE TOSE, July 2002. http://people.cis.ksu.edu/~dag/740fall02/materials/740f02presentations22.pdf

[Ionita 2002] Ionita, Mugurel T.; Hammer, Dieter K.; & Obbink, Henk. "Scenario-Based Software Architecture Evaluation Methods: An Overview." In Workshop on Methods and Techniques for Software Architecture Review and Assessment at the *International Conference on Software Engineering*. Orlando, FL, May 19-25, 2002. http://www.win.tue.nl/oas/architecting/aimes/papers/Scenario-Based%20SWA%20Evaluation%20Methods.pdf

[Jones 2006] Jones, C. *The Economics of Software Maintenance in the Twenty First Century*. 2006. http://www.compaid.com/caiinternet/ezine/capersjones-maintenance.pdf

[Kazman 1994] Kazman, Rick; Bass, Len; Webb, Mike; & Abowd, Gregory. "SAAM: A Method for Analyzing the Properties of Software Architectures," 81-90. *Proceedings of the 16th International Conference on Software Engineering.* ICSE 94, Sorrento, Italy, May 16-21, 1994. IEEE Computer Society Press, 1994.

[Kazman 1996] Kazman, R.; Abowd, G.; Bass, L.; & Clements, P. "Scenario-Based Analysis of Software Architecture." *IEEE Software 13,* 6, (November 1996): 47-55.

[Kruchten 2001] Kruchten, P.; Obbink, H.; Kozaczynski, W.; Posterna, H.; et al. *Software Architecture Review and Assessment (SARA) Report.* SARA W. G., 2001. http://pkruchten.files.wordpress.com/2011/09/sarav1.pdf


59 See http://www.sei.cmu.edu/training/find/courses.cfm?category=Software%20Architecture

[Pittman 1990] Pittman, J. & Kitrick, C. “VUIMS: A Visual User Interface Management System,” 36-46. *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology*, Snowbird, Utah, October 1990.

[SEI 2009] *SEI Monitor*, January 2009. Software Engineering Institute, Carnegie Mellon University, 2009. http://www.sei.cmu.edu/saturn/2009/Monitor_Jan_09_Small2.pdf

[Szekely 1989] Szekely, P. “Standardizing the Interface Between Applications and UIMSS,” 34-42. *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology*. Williamsburg, Virginia, November 13-15, 1989. ACM, 1989.

## 7.5 Quality Attributes

### 7.5.1 The Challenge: Meet the Non-Functional Requirements of Software Systems

In the early 1990s, there did not exist a structured and repeatable way of analyzing tradeoffs among the non-functional requirements, or quality attributes, of a system.[60] Analysis of such tradeoffs was ad hoc and intuitive, and there was not a way for system developers to explain to other stakeholders in their organizations why they thought their proposed systems were appropriate or superior to other system designs or architectures for achieving desired quality attributes. This problem for software developers was articulated by Boehm as early as 1978 [Boehm 1978]:

> *Finally, we concluded that calculating and understanding the value of a single overall metric for software quality may be more trouble than it is worth. The major problem is that many of the individual characteristics of quality are in conflict; added efficiency is often purchased at the price of portability, accuracy, understandability, and maintainability; added accuracy often conflicts with portability via dependence on word size; conciseness can conflict with legibility. Users generally find it difficult to quantify their preferences in such conflict situations.*

### 7.5.2 A Solution: Focus on Tradeoffs Among Quality Attributes

The SEI recognized, in the early 1990s, that architecture-like thinking was evolving in various SEI efforts and decided to devote attention to the emerging discipline of architecture-centric engineering by creating a focus on software architecture. The idea that quality attributes influence the shape of the architecture, and that the architecture is fundamental to the system, emerged from SEI work in rate monotonic analysis (RMA). Initially, SEI staff members worked on the theory of scheduling, and on making scheduling theory practical by modifying runtime systems and compilers. Later, this work led to the realization that scheduling theory could be used to analyze systems as well as to build them—that the structure of a system can be looked at critically from the point of view of real-time scheduling, yielding valuable insights about predicted system behavior. In this way, SEI work in real-time systems evolved from scheduling to analysis. The notion of considering systems from an analytic point of view, and how such a consideration gives insight into structure, was the key insight that emerged from RMA; the analytical framework that RMA provided led to structuring principles for how to design and analyze real-time systems. By analogy, SEI staff members realized that such a framework could be applied to every other important quality attribute, leading to the conclusion that quality attributes are the dominant influence on architecture.

From the early days of this work, SEI staff members decided that architecture is about making tradeoffs among various quality attributes, including performance, security, modifiability, reliability, and usability, system qualities that cannot be achieved without a unifying architectural vision. Architecture came to be seen and used as an artifact for early analysis to make sure that a design approach will yield an acceptable system. SEI researchers also concluded that quality attributes are the dominant influence on scheduling, and on making scheduling theory practical by modifying runtime sys-

60 Non-functional requirements describe characteristics (qualities) of a system—that is, how it should be—in contrast to functional requirements, which describe behaviors (functions) of a system—what it should do.

tems and compilers. Later, this work led to the realization that scheduling theory could be used to analyze systems as well as to build them—that the structure of a system can be looked at critically from the point of view of real-time scheduling, yielding valuable insights about predicted system behavior.

Developing systematic ways to relate the software quality attributes of a system to the system's architecture provides a sound basis for making objective decisions about design tradeoffs and enables engineers to make reasonably accurate predictions about a system's attributes that are free from bias and hidden assumptions. The ultimate goal was the ability to quantitatively evaluate and trade off multiple software quality attributes to arrive at a better overall system [Barbacci 1995].

Toward this goal, the SEI developed the Architecture Tradeoff Analysis Method (ATAM), described in the next section, and the Quality Attribute Workshops (QAWs) [Barbacci 2003a], a method for identifying architecture-critical quality attributes before there is a software architecture to which the ATAM can be applied. Since the development of these methods, a focus on quality attributes has been a consistent theme and emphasis of SEI work in software architecture.

**The View from Others**

*Over the last 15 years, the SEI has become a sort of mecca for software architecture—a place where anyone who is doing any work related to software architecture must go."*

– Phillippe Kruchten, University of British Columbia and developer of the Rational Unified Process (RUP) [SEI 2008]

### 7.5.3 The Consequence: Quality Attributes Reliably Identified, Added to Specifications

SEI staff members tested and validated this insight into the primacy of quality attributes in their early experiences conducting architecture evaluations. Whether they were evaluating a financial system or an avionics system, conversant in the domains but not expert, they succeeded in finding risks by evaluating the systems from the point of view of different quality attributes. This experience validated the idea that modifiability principles, real-time principles, or reliability principles apply independently of domain—whether the system being evaluated is a car or an aircraft.

### 7.5.4 The SEI Contribution

A lasting influence of the SEI work in the field of software architecture and software development can be seen in the pervasive attention paid to quality attributes and a general acknowledgment that requirements specifications need to include them. Questions of how secure, timely, reliable, and usable systems must be are now fundamental components of the processes used in all software development projects.

### 7.5.5 References

[Abowd 1996] Abowd, Gregory; Bass, Len; Clements, Paul; Kazman, Rick; Northrop, Linda; & Zaremski, Amy. *Recommended Best Industrial Practice for Software Architecture Evaluation* (CMU/SEI-96-TR-025). Software Engineering Institute, Carnegie Mellon University, 1997. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=12653

[Barbacci 1995] Barbacci, Mario; Klein, Mark; Longstaff, Thomas; & Weinstock, Charles. *Quality Attributes* (CMU/SEI-95-TR-021). Software Engineering Institute, Carnegie Mellon University, 1995. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=12433

[Barbacci 2003a] Barbacci, Mario; Ellison, Robert; Lattanze, Anthony; Stafford, Judith; Weinstock, Charles; & Wood, William. *Quality Attribute Workshops (QAWs), 3rd Edition* (CMU/SEI-2003-TR-016). Software Engineering Institute, Carnegie Mellon University, 2003. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=6687

[Barbacci 2003b] Barbacci, Mario; Clements, Paul; Lattanze, Anthony; Northrop, Linda; & Wood, William. *Using the Architecture Tradeoff Analysis Method (ATAM) to Evaluate the Software Architecture for a Product Line of Avionics Systems: A Case Study* (CMU/SEI-2003-TN-012). Software Engineering Institute, Carnegie Mellon University, 2003. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=6447

[Boehm 1978] Boehm, B.; Brown, J. R.; Kaspar, J. R.; Lipow, M. L.; & MacCleod, G. *Characteristics of Software Quality*. American Elsevier, 1978 (ISBN 0444851054).

[Clements 2005] Clements, Paul; Bergey, John; & Mason, David. *Using the SEI Architecture Tradeoff Analysis Method to Evaluate WIN-T: A Case Study* (CMU/SEI-2005-TN-027). Software Engineering Institute, Carnegie Mellon University, 2005. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=7467

[Gallagher 2000] Gallagher, Brian. *Using the Architecture Tradeoff Analysis Method to Evaluate a Reference Architecture: A Case Study* (CMU/SEI-2000-TN-007). Software Engineering Institute, Carnegie Mellon University, 2000. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=5109

[Jones 2001] Jones, Lawrence & Lattanze, Anthony. *Using the Architecture Tradeoff Analysis Method to Evaluate a Wargame Simulation System: A Case Study* (CMU/SEI-2001-TN-022). Software Engineering Institute, Carnegie Mellon University, 2001. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=5535

[Kazman 1994] Kazman, Rick; Bass, Leonard J.; & Webb, Mike. “SAAM: A Method for Analyzing the Properties of Software Architectures,” 81-90. *Proceedings of the 16th International Conference on Software Engineering.* ICSE, Sorrento, Italy, May 16–21, 1994. IEEE Computer Society Press, 1994

[Kazman 1996] Kazman, R.; Abowd, G.; Bass, L.; & Clements, P. “Scenario-Based Analysis of Software Architecture.” *IEEE Software 13*, 6 (November 1996): 47-55.

[SEI 2008] SEI *News Items.* “Fourth Annual SATURN Workshop Attracts International Software Architecture Practitioners.” June 6, 2008. http://www.sei.cmu.edu/newsitems/saturn08post.cfm

[SEI 2009a] Software Engineering Institute. “Army Commitment to Strategic Software Improvement Grows,” 17. *2008 Year in Review*. Software Engineering Institute, Carnegie Mellon University, 2009.

[SEI 2009b] *SEI Monitor, January 2009*. Software Engineering Institute, Carnegie Mellon University, 2009. http://www.sei.cmu.edu/saturn/2009/Monitor_Jan_09_Small2.pdf

## 7.6 Architecture Tradeoff Analysis Method

### 7.6.1 The Challenge: Determining the Best Architectural Design for Defense Systems

Most complex software systems are required to be modifiable and have good performance. They may also need to be secure, interoperable, portable, and reliable. But for any particular system, what precisely do these quality attributes mean—modifiability, security, performance, reliability? The achievement of quality attributes such as these in a software system depends more on the software architecture than on code-related issues: language choice, fine-grained design, algorithms, data structures, testing, and so forth. Software architectures are complex and involve many design tradeoffs. Without undertaking a formal analysis process, an organization developing a system cannot ensure that the architectural decisions made—particularly those that affect the achievement of particular quality attributes—are advisable ones that appropriately mitigate risks.

In the early 1990s, there was not a structured and repeatable way of analyzing tradeoffs among quality attributes. Analysis of tradeoffs in quality attributes was ad hoc and intuitive, and there was not a way for architects to explain to other stakeholders in their organizations why they thought their proposed architectures were appropriate or superior to other possible architectures.

### 7.6.2 A Solution: The Architecture Tradeoff Analysis Method

In an earlier effort, the SEI developed the Software Architecture Analysis Method (SAAM) as a way of looking at modifiability [Kazman 1994, 1996]. In considering modifiability, SEI researchers realized that there were other quality attributes that were important, and that there are tradeoffs among them. The Architecture Tradeoff Analysis Method (ATAM) emerged from the SAAM as a way of analyzing these other quality attributes [Kazman 2000].

The purpose of the ATAM is to assess the consequences of architectural decisions in light of quality attribute requirements and business goals. An ATAM is an early lifecycle analysis method that is designed to discover risks in decisions that might create future problems in some quality attribute and to discover tradeoffs in decisions affecting more than one quality attribute. Discovered risks can then be made the focus of mitigation activities (further design, further analysis, or prototyping), and surfaced tradeoffs can be explicitly identified and documented.

An evaluation using the ATAM typically takes three to four days and gathers together a trained evaluation team, architects, and representatives of the architecture's various stakeholders. The ATAM consists of nine steps, which include a discussion of business drivers and architecture, and architectural approaches, among other topics. The output of an ATAM is a presentation and/or a written report that includes the major findings of the evaluation. These are typically a set of architectural approaches identified; a "utility tree"—a hierarchic model of the driving architectural requirements; the set of scenarios generated and the subset that were mapped onto the architecture; quality-attribute-specific questions that were applied to the architecture and the responses to these questions; a set of identified risks; a set of identified non-risks; and synthesis of the risks into risk themes that threaten to undermine the business goals for the system. The most important results are improved architectures.

### 7.6.3 The Consequence: Effective Evaluation of Architecture Designs

Building on the SAAM, the ATAM represents a further evolutionary development in the use of scenarios to perform architecture evaluation. Major defense contractors, such as Boeing and Raytheon, now have architecture evaluation teams and architecture evaluation as an element of their architect certification processes. The U.S. Army has reported that use of the ATAM and other scenario-based architecture evaluation methods reduced risk in schedule and cost, improved documentation, and resulted in a higher quality product [SEI 2010]. Both General Motors and the Air Force Space Surveillance Network Model (SSNAM) used the results of SEI-conducted ATAM evaluations to inform crucial decisions about system-evolution paths.

Many published case studies document positive results from the use of the ATAM to evaluate software architectures, including architectures for the U.S. Army's Warfighter Information Network-Tactical (WIN-T) system [Clements 2005]; the Common Avionics Architecture System (CAAS) for a family of U.S. Army Special Operations helicopters [Barbacci 2003]; the Wargame 2000 system at the Joint National Integration Center (formerly known as the Joint National Test Facility [JNTF]) in Colorado [Jones 2001]; and a reference architecture for ground-based command-and-control systems [Gallagher 2000].

The SEI completed a study of the impact of the ASSIP-sponsored Quality Attribute Workshops (QAWs) and architecture evaluations using the ATAM. Ten of the eleven programs responding to the survey indicated that the ATAM/QAW produced better results than they traditionally obtained. Six of the ten indicated that the ATAM/QAW costs no more than their traditional approaches. The study also provided evidence of the potential for the ATAM/QAW having even more impact if done "proactively."

**The View from Others**

*The year [2008] also saw a scaling up of the Army's interest in learning and applying the SEI's software architecture knowledge through ASSIP. A concerted effort conducted through the SEI helped the Army grow its ranks of software experts trained in the SEI Architecture Tradeoff Analysis Method (ATAM). Army personnel have taken part in about a dozen ATAM evaluations to date. The Army has also seen an added, immediate benefit from the architecture training: The PEOs have used them to reveal software risks early in projects' lifetimes.*

[SEI 2009, p. 17]

*It's a towering accomplishment to get everyone's heads wrapped around this and get all of these important issues distilled in this amount of time.*

– General Motors staff
on the ATAM

### 7.6.4 The SEI Contribution

The development of the ATAM and its ultimate codification as a methodology for dealing with tradeoffs among quality attributes, was influenced by a 1996 workshop that the SEI conducted. Participants from the Georgia Institute of Technology, Motorola, IBM T.J. Watson Research, Carnegie Mellon University, Siemens, Nortel, Rational, AT&T Bell Laboratories, and Performance Engineering Services met with SEI staff members to discuss and validate an SEI report on best practices in architecture evaluation [Abowd 1996]. The workshop helped to shape the structure of the evaluation method that evolved into the ATAM, which includes the Quality Attribute Workshop. The SEI also published case studies documenting the use of QAWs in source selection for a

DoD system acquisition [Bergey 2002] and as a tool to evaluate architectural design approaches in a major system acquisition [Bergey 2000].

A measure of the influence of the SEI architecture methods and techniques is that they have been incorporated into a handbook for practitioners on the practice of architecting written by an Open Group Master Certified IT architect [Eeles 2010]. The book advises practitioners on the importance of quality attributes and refers the reader to SEI techniques and methods (including quality attribute scenarios, architectural tactics, attribute-driven design, and ATAM). Additionally, of the scholarly articles on the ATAM, the three most-popular [Kazman 1998, 1999, 2001] were cited, oldest to more recent, 490, 198, and 153 times.

Many architecture evaluation methods emerged after the SEI published and disseminated the ATAM, and most acknowledge its influence. Bosch, Boeing, and Raytheon have publicly acknowledged their adoption of the ATAM and the strong influence the ATAM has had on their approach to developing software-reliant systems. The ATAM is generally acknowledged today to be the leading method in the area of software architecture evaluation.

### 7.6.5 References

[Abowd 1996] Abowd, Gregory; Bass, Len; Clements, Paul; Kazman, Rick; Northrop, Linda; & Zaremski, Amy. *Recommended Best Industrial Practice for Software Architecture Evaluation* (CMU/SEI-96-TR-025). Software Engineering Institute, Carnegie Mellon University, 1997. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=12653

[Barbacci 2003] Barbacci, Mario; Clements, Paul; Lattanze, Anthony; Northrop, Linda; & Wood, William. *Using the Architecture Tradeoff Analysis Method (ATAM) to Evaluate the Software Architecture for a Product Line of Avionics Systems: A Case Study* (CMU/SEI-2003-TN-012). Software Engineering Institute, Carnegie Mellon University, 2003. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=6447

[Bergey 2000] Bergey, John; Barbacci, Mario; & Wood, William. *Using Quality Attribute Workshops to Evaluate Architectural Design Approaches in a Major System Acquisition: A Case Study* (CMU/SEI-2000-TN-010). Software Engineering Institute, Carnegie Mellon University, 2000. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=5123

[Bergey 2002] Bergey, John & Wood, William. *Use of Quality Attribute Workshops (QAWs) in Source Selection for a DoD System Acquisition: A Case Study* (CMU/SEI-2002-TN-013). Software Engineering Institute, Carnegie Mellon University, 2002. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=5933

[Clements 2005] Clements, Paul; Bergey, John; & Mason, David. *Using the SEI Architecture Tradeoff Analysis Method to Evaluate WIN-T: A Case Study* (CMU/SEI-2005-TN-027). Software Engineering Institute, Carnegie Mellon University, 2005. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=7467

[Eeles 2010] Eeles, Peter & Cripps, Peter. *The Process of Software Architecting*. Addison-Wesley Professional, 2010 (ISBN 0321357485).

[Gallagher 2000] Gallagher, Brian. *Using the Architecture Tradeoff Analysis Method to Evaluate a Reference Architecture: A Case Study* (CMU/SEI-2000-TN-007). Software Engineering Institute, Carnegie Mellon University, 2000. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=5109

[Jones 2001] Jones, Lawrence & Lattanze, Anthony. *Using the Architecture Tradeoff Analysis Method to Evaluate a Wargame Simulation System: A Case Study* (CMU/SEI-2001-TN-022). Software Engineering Institute, Carnegie Mellon University, 2001. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=5535

[Kazman 1994] Kazman, R.; Abowd, G.; Bass, L.; & Webb, M. "SAAM: A Method for Analyzing the Properties of Software Architectures," 81-90. *Proceedings of the 16th International Conference on Software Engineering.* ICSE 16, Sorrento, Italy, May 1994.

[Kazman 1996] Kazman, R.; Abowd, G.; Bass, L.; & Clements, P. "Scenario-Based Analysis of Software Architecture." *IEEE Software 13,* 6 (November 1996): 47-55.

[Kazman 1998] Kazman, R.; Klein, M.; Barbacci, M.; Longstaff, T.; Lipson, H. & Carriere, J. "The Architecture Tradeoff Analysis Method," 68-69. *Proceedings of the 4th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 98*). Monterey, CA, August 10-14, 1998. IEEE Computer Society, 1998.

[Kazman 1999] Kazman, Rick; Barbacci, Mario; Klein, Mark; Carriere, S. Jeremy; & Woods, Steven G. "Experience with Performing Architecture Tradeoff Analysis," 54-63. *Proceedings of the International Conference on Software Engineering (ICSE).* Los Angeles, CA, May 16-22, 1999. IEEE, 1999.

[Kazman 2000] Kazman, Rick; Klein, Mark; & Clements, Paul. *ATAM: Method for Architecture Evaluation* (CMU/SEI-2000-TR-004). Software Engineering Institute, Carnegie Mellon University, 2000. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=5177

[Kazman 2001] Kazman, R.; Asundi J.; & Klein, M. "Quantifying the Costs and Benefits of Architectural Decisions," 297-306. *Proceedings of the 23rd International Conference on Software Engineering (ICSE 2001)*. May 12-19, 2001. IEEE Computer Society, 2001.

[SEI 2009] Software Engineering Institute. "Army Commitment to Strategic Software Improvement Grows," 17. *2008 Year in Review*. Software Engineering Institute, Carnegie Mellon University, 2009.

[SEI 2010] Software Engineering Institute. "Army Requires PEOs to Appoint Chief Software Architect," 20. *2009 Year in Review*. Software Engineering Institute, Carnegie Mellon University, 2010.

## 7.7 Feature-Oriented Domain Analysis

### 7.7.1 The Challenge: Achieve Cost Reductions By Means of Software Reuse

A 1993 report by the Government Accounting Office[61] [GAO 1993] cited a comprehensive effort to incorporate software reuse practices into the DoD's software development efforts. Reuse was viewed as a way to manage software costs, which in 1993 exceeded $24 billion a year, and to improve the DoD's ability to develop and maintain high-quality software. In practice, however, the promise of software reuse had not been realized. The report stated that "methodologies to implement reuse have not been fully developed, tools to support a reuse process are lacking, and standards to guide critical software reuse activities have not been established" [GAO 1993, p. 1].

One of the technical issues cited in the GAO report was that the variation in the domains in which software reuse was being attempted was proving intractable. Standard methods were needed to process information about domains—"domain analysis"—the purpose of which is "to generalize common features in similar application areas, identify the common objects and operations in these areas, and define and describe their relationships."

**The View from Others**

An examination of the proceedings from the Software Product Lines Conference (SPLC) over its 16-year history shows a preponderance of papers that refer to FODA and domain analysis. FODA has proven to be a fruitful research area, and FODA and variation/variability analysis have been the most common themes at SPLC.

### 7.7.2 A Solution: Feature-Oriented Domain Analysis

At this time, the SEI was working with an Army program called Army Tactical Command and Control System (ATCCS) [Williamson 2005] that had five separate programs supporting five function areas: maneuver control, air defense, combat service support, intelligence/electronic warfare, and fire support. Seeking to identify isolated areas with common requirements shared by these programs, the SEI identified "domains," or specific areas of knowledge. An SEI staff member on this project proposed the idea of "features" based on experience at AT&T, where equipment builds were classified by their features. For example, for switches, some would be automatic, some would be manual, etc. Features would be the way to analyze the common characteristics of a domain, prior to formal requirements engineering. The SEI pioneered this idea of features and developed a methodology that it called "feature-oriented domain analysis," or FODA [Kang 1990].

The SEI saw a domain as a special area of knowledge, and characterized large-scale systems as comprising several domains. The Army asked the SEI to look at movement control—route planning, scheduling, and deconfliction, to ensure that, for example, a road would not be taken up by two different vehicles at the same time. Movement control was attractive to the Army because it is a domain that applied to many of the ATCCS systems. The SEI described movement control as a domain, identifying commonalities about how units were handled and organized, how routes were planned and convoys scheduled, etc. After conducting this domain analysis, the SEI developed a

61 Now known as the Government Accountability Office.

design with some variation built in and did an implementation for the Army, building a system that could operate on Army equipment.

### 7.7.3 The Consequence: Application to Reuse and a Product Lines Approach

Because of the resistance of contractors and the inability of program managers to justify the effort with ongoing programs, the results of this work were not transitioned to an operational Army system. However, the concepts developed were picked up in the reuse and eventually the product lines communities. The key insight was that effective reuse requires an understanding of the commonalities across all systems in a particular mission area, such as command and control, and where they vary, this information is captured in a model representation. In FODA, the information was captured in tree fashion with a hierarchy of mandatory features, optional features, and alternative ways of describing how the features would be used.

The SEI began to hold product line workshops around 1996, and the work that had begun with FODA evolved into a concentration on product lines. At this time, the SEI began working of these issues with industry customers, including Motorola, HP, Avaya (Lucent at the time), and other early adopters.

### 7.7.4 The SEI Contribution

Compared to other approaches to domain analysis, the SEI's FODA was a lightweight approach that could be applied in two to three weeks. The DARPA STARS program also developed a domain analysis approach, but it required more effort and was more time consuming. The SEI advocated moving quickly from FODA on to requirements, architecture, and implementation, preferring that domain analysis feed downstream activities. This methodology evolved into what is now known as commonality/variation design, and then architecture-component variation points, in current product line practice.

Others at the time were working on object-oriented design, building a set of classes and a class hierarchy once and then having everyone in the software development organization use these classes for all the systems they built. Object-oriented design, the principal mechanism at the time for achieving reuse, was undermined by variation and an inability by organizations to control the proliferation of subclasses, which effectively prevented widespread reuse. As the product line concept began to mature, the software community began to recognize the need for domain analysis.

Many other systems and methods have been built around the FODA model. David Weiss and Chi Tau Robert Lai wrote a textbook in 1999 called *Software Product Line Engineering: A Family-Based Software Development Process* [Weiss 1999]. The methodology described therein, known as commonality variability analysis, examined, for example, organizational commonality in provisioning, configuration management, and billing operations. It was broader than domain analysis, but similar, and there were cross-influences between Weiss and the SEI.

In 2000, Krzysztof Czarnecki [Czarnecki 2000] worked on automatic software generation that was built on FODA models or variation models. Around the same time, in early 2002, a company in Germany called Pure Systems developed a tool to manage variation and feature modeling. Pure Systems had been working in configuration management and moved into feature modeling. Also around that time, Charles Kreuger of Big Lever developed Gears, which had built into it the feature concept. Gears also applied to configuration management and mass customization—a lot of

small components could be integrated in different ways, with integration guided by what things were common, how the common pieces fit together, and what were the variations.

The principal contribution of the SEI in developing FODA was to demonstrate that to achieve reuse, simply identifying common elements and features is insufficient. To succeed, reuse requires a more systematic identification of commonality and variation, how things vary, and where they vary; and then variation must be planned in advance, anticipating and planning for changes that are likely to occur. Managing variation was the main emphasis in FODA that distinguished it from other methodologies.

The preponderance of tools that perform domain analysis (80-90 percent) use FODA as a starting point. They all use the alternative and optional kinds of features and may add information that the SEI did not account for, but all are built around the model that the SEI developed.

### 7.7.5 References

[Czarnecki 2000] Czarnecki, K. & Eisenecker, U. *Generative Programming: Methods, Tools, and Applications*. Addison-Wesley, 2000 (ISBN 0201309777).

[GAO 1993] General Accounting Office. *Software: Major Issues Need to Be Resolved Before Benefits Can Be Achieved* (GAO/IMTEC-93-16). General Accounting Office, 1993.

[Kang 1990] Kang, Kyo; Cohen, Sholom; Hess, James; Novak, William; & Peterson, A. *Feature-Oriented Domain Analysis (FODA) Feasibility Study* (CMU/SEI-90-TR-021). Software Engineering Institute, Carnegie Mellon University, 1990. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=11231

[Weiss 1999] Weiss, David M. & Lai, Chi Tau Robert. *Software Product Line Engineering: A Family-Based Software Development Process.* Addison-Wesley Professional, 1999 (ISBN 0201694387).

[Williamson 2005] Williamson, John. “Army Tactical Command and Control System (ATCCS) (United States), Systems,” 673. *Jane’s Military Communications 2005–06, 26th Edition*. John Williamson, ed. Janes Information Group, 2005 (ISBN 0710626991).

## 7.8 Software Product Lines

### 7.8.1 The Challenge: Achieve Reuse That Pays

Software reuse has been a topic of interest and discussion since the first conference on software engineering that was held by NATO in Garmisch, Germany, in 1968 [McIlroy 1969]. Most organizations build families of similar systems and so it makes sense to pursue economies of scale. Early forays into reuse were confined to code, functions, and procedures. Later, the development of the Ada programming language and the wave of object-oriented languages provided the ability to reuse packages of functionality and software components.

However, experience proved that the reuse of code-level elements did not provide the desired economic advantages; such reuse did little to reduce the principal sources of expense in software development. The major expense of software development lies in the process of defining requirements and structuring a solution, as well as in all the other non-code artifacts such as test cases, business artifacts, and the architecture that defines how software elements fit together. To achieve reuse that provides substantial return on investment—reuse that pays, or strategic reuse—a comprehensive approach was needed that would involve more than just reusing pieces of code or serendipitously discovering artifacts in a library to reuse.

### 7.8.2 A Solution: Software Product Lines

Predominantly, strategic reuse capitalizes on commonality—common features—and manages variation. When multiple similar products are developed, there is some degree to which they are the same, but also some degree to which they vary. Economic advantage is achieved through a systematic product line approach that effectively manages this variation. A software product line is "a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way" [Clements 2001]. Creating a software product line depends on establishing a software architecture, or product line architecture, for the entire set of systems; and software architecture was only beginning to receive attention at that time.

By the 1980s, diverse areas, such as automobiles, aircraft, machine tools, and, more recently, computer hardware, were using the concept of a product line; but applying that concept to a software product line was not common practice. The SEI was influenced to begin its formal investigation into software product lines by a number of related SEI experiences and by DoD and commercial attempts to create software product lines in the late 1980s and early 1990s. The SEI had developed a reference architecture for the B-2 simulator that was later applied to other simulators. The DARPA STARS program was experimenting with the development of software product lines [STARS 1983]. AFSC/ESD initiated the PRISM project to experiment with the definition of a product line approach to the development of command centers. There were also a number of commercial industry efforts to define a software product line, most notably by CelsiusTech Systems AB.

The SEI launched a full-scale investigation into software product lines, the Product Line Practice Initiative. The SEI vision was that product line development would become a low-risk, high-return proposition for all organizations—techniques for finding and exploiting system commonalities and for controlling variability would be standard practice in the DoD, government, and industry. The SEI aimed to facilitate and accelerate that transformation. The SEI strategy was to first

understand the basis for product line success that was starting to occur in the commercial space, and then to build on that innovation and, ultimately, transition knowledge and results to the DoD.

In 1996, the SEI convened the first of what was to become a series of workshops to learn more about the state of product line practices. Organizations attending the first workshop included CelsiusTech, Robert Bosch GmBH, Schlumberger, Philips Laboratories, Caterpillar, and the U.S. Army [Bass 1997, 1998, 1999]. The SEI became an active participant in a similar set of workshops occurring in Europe that were funded by the European Union as part of research projects focused on product families. In 1998, the SEI held the first of a series of DoD Product Line Workshops to share successful product line practices and identify DoD-specific challenges to strategic reuse [Bergey 1998].[62]

**The View from Others**

*COM SEC will benefit greatly from the growth in technology awareness and process improvement resulting from the support provided by the SEI.*

– Advanced Multiplex Test System, CECMOM Army

The SEI also engaged directly with DoD organizations that were attempting product line efforts. The National Reconnaissance Office (NRO), the Joint National Test Facility, the U.S. Army Special Operations Aviation Technical Application Program Office, the F-22 Pilot Training Program, and the Army's Force XXI Battle Command Brigade and Below (*FBCB2*) Program were among the many to be successful. The NRO's Control Channel Toolkit (CCT) was a software asset base for ground-based spacecraft command-and-control systems that was completed on schedule and within budget, with no outstanding risks or actions. The first system in this product line experienced a 50 percent reduction in overall cost and schedule and nearly tenfold reductions in development personnel and defects [Clements 2001, Section 10].

Key contributions by the SEI to the discipline of product line practice, created to support the adoption of software product lines, include the following:

- **Framework for Software Product Line Practice**. Stemming from the SEI's immersion in the product line community, its applied research, collaborations with DoD and commercial organizations, and participation in workshops, and conferences, the SEI developed the web-based Framework for Software Product Line Practice as a reference model that describes the technical, management, and business practices essential for software product lines. The framework was intended to be a complete and thorough reference model for every aspect of product line practice, as opposed to a development model or roadmap. The last revision of the framework took place in 2006. There was also an Acquisition Companion to the framework that described the practices from the view of a government acquisition organization.
- ***Software Product Lines: Practices and Patterns*** [Clements 2001]. This was the first book on software product lines, and includes the framework, product line case studies, and a set of patterns that aimed to help organizations achieve results with the practices.

62 Reports on subsequent workshops can be found in the digital library on the SEI website (http://resources.sei.cmu.edu/library/). All were written by Bergey as primary author except one, by S. Cohen.

- **A software product line curriculum.** A series of four courses, an executive seminar, and a product line acquisition tutorial were developed and delivered to more than 3,500 individuals from major defense contractors, the DoD, and commercial organizations.
- **Software Product Line Conference (SPLC).** Begun by the SEI in 2000, when more than 100 people attended, SPLC merged in 2004 with its European counterpart, the Product Family Engineering Workshop. Since then, SPLC has always attracted an international audience of 150-200 researchers and practitioners.
- **Structured Intuitive Model for Product Line Economics (SIMPLE).** This comprehensive cost model for software product lines aids the development of a business case.
- **Product Line Analysis.** This approach combined domain understanding and requirements engineering for product lines to identify opportunities for large-grained reuse.
- **Product Line Technical Probe (PLTP).** This method supports gathering information about an organization's readiness to adopt or ability to succeed with a product line approach. The **Product Line Quick Look (PLQL)** was a lighter weight version.
- **Product Line Production Plan** [Chastek 2002]. This describes how to specify the technical approach for how core assets are to be used to develop a product in a product line.
- **Product Line Adoption Roadmap** [Northrop 2004]. This provides a generic roadmap to guide a manageable, phased product line adoption strategy.

### 7.8.3 The Consequences: Product Line Use Expands and Provides Benefits

Companies such as Cummins, Nokia, Hewlett Packard, Philips, Robert Bosch, Raytheon, General Motors, NCR, Intuit, ABB, Siemens, Panasonic, Boeing, Rockwell Collins, General Dynamics, and Northrop Grumman are among a growing number of organizations using SEI materials and methods to support their product line practices. They report benefits that include 60 percent cost reduction, decreased time to market by as much as 98 percent as compared with their past experience, and the ability to move into new markets in months rather than years [Pohl 2005, Clements 2001].

Defense examples include the Army Training Support Center (ATSC), Advanced Multiplex Test System, Army's Common Avionics Architecture System (CAAS) Product Line, Textron Overwatch Intelligence Center Software Product Line, the Live, Virtual, Constructive Integrating Architecture (LVCIA) product lines at Army PEO/STRI, and BAE's Diamond software product line [Jones 2010].

### 7.8.4 The SEI Contribution

As with SEI efforts in other areas, the SEI was not the only contributor to the notions of software product lines, but was an early leader and a significant contributor to the discipline and maturation of the practice. The international engagement with and continued success of the Software Product Line Conference, as well as the pervasive use of software product line practices in government and industry is evidence of the continuing impact of this SEI work.

### 7.8.5 References

[Bass 1997] Bass, Len; Clements, Paul; Cohen, Sholom; Northrop, Linda; & Withey, James. *Product Line Practice Workshop Report* (CMU/SEI-97-TR-003). Software Engineering Institute,

Carnegie Mellon University, 1997. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=12815

[Bass 1998] Bass, Len; Chastek, Gary; Clements, Paul; Northrop, Linda; Smith, Dennis; & Withey, James. *Second Product Line Practice Workshop Report* (CMU/SEI-98-TR-015). Software Engineering Institute, Carnegie Mellon University, 1998. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=13147

[Bass 1999] Bass, Len; Campbell, Grady; Clements, Paul; Northrop, Linda; & Smith, Dennis. *Third Product Line Practice Workshop Report* (CMU/SEI-99-TR-003). Software Engineering Institute, Carnegie Mellon University, 1999. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=13347

[Bergey 1998] Bergey, John; Krut, Jr., Robert; Clements, Paul; Cohen, Sholom; Donohoe, Patrick; Jones, Lawrence; Northrop, Linda; Tilley, Scott; Smith, Dennis; & Withey, James. *DoD Product Line Practice Workshop Report* (CMU/SEI-98-TR-007). Software Engineering Institute, Carnegie Mellon University, 1998. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=13069

[Chastek 2002] Chastek, Gary; & McGregor, John. *Guidelines for Developing a Product Line Production* Plan (CMU/SEI-2002-TR-006). Software Engineering Institute, Carnegie Mellon University, 2002. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=6067

[Clements 2001] Clements, Paul; & Northrop, Linda. *Software Product Lines: Practices and Patterns*. Addison-Wesley, 2001 (ISBN 0201703327).

[Kang 1990] Kang, K.; Cohen, S.; Hess, J.; Novak, W.; & Peterson, A. S. *Feature-Oriented Domain Analysis (FODA) Feasibility Study* (CMU/SEI-90-TR-021). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1990.

[Jones 2010] Jones, L.; & Northrop, L., "Recent Experiences with Software Product Lines in the US Department of Defense." Software Product Line Conference 2010, Jeju Island, South Korea, September 13-17, 2010.

[McIlroy 1969] McIlroy, Malcolm Douglas. "Mass Produced Software Components," 29. S*oftware Engineering: Report of a Conference Sponsored by the NATO Science Committee*, Garmisch, Germany, 7-11 October 1968. Scientific Affairs Division, NATO, 1969.

[Northrop 2004] Northrop, Linda. *Software Product Line Adoption Roadmap* (CMU/SEI-2004-TR-022). Software Engineering Institute, Carnegie Mellon University, 2004. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=7237

[Pohl 2005] Pohl, Klaus; Bockle, Gunter; & van der Linden, Frank. *Software Product Line Engineering: Foundations, Principles, and Techniques*. Springer, 2005 (ISBN 3540243720).

[STARS 1983] "Software Technology for Adaptable, Reliable Systems (STARS) Program Strategy." *ACM SIGSOFT Software Engineering Notes 8*, 2 (April 1983): 56-108 (DOI 10.1145/1005959.1005966). http://doi.acm.org/10.1145/1005959.1005966

# 8 Forensics

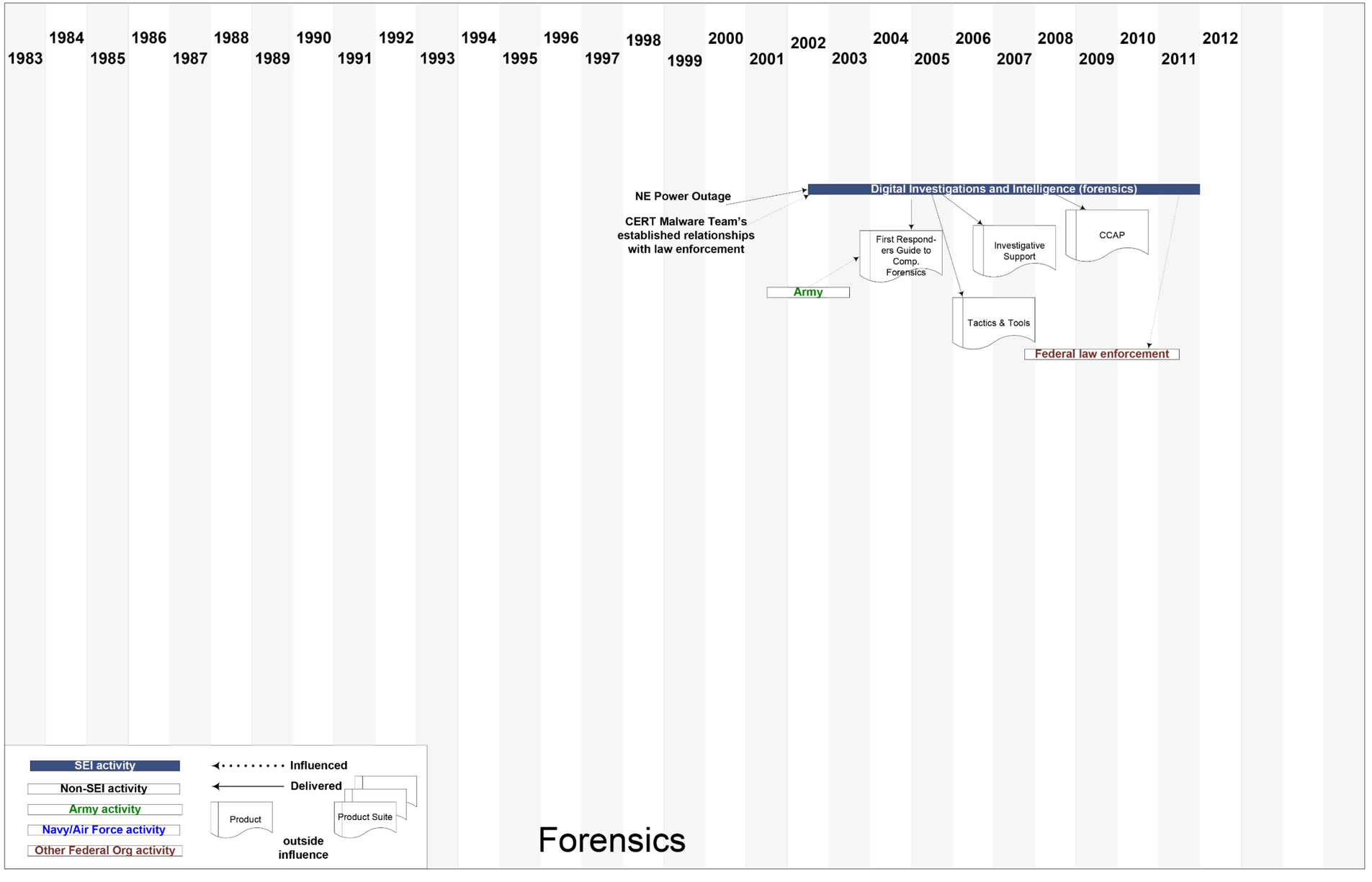


*Figure 8: Forensics Timeline*

# 8.0 Introduction to Computer Forensics: Digital Intelligence and Investigation

In the early days of the internet, protection against cyber attacks was not a high priority for the researchers using it. Cybersecurity became more important with the expansion into commercial and international internet use. At the same time, attacks became more sophisticated and the demand for cyber forensics experts increased. In the 1980s and 1990s, computers were the objects of crimes, and investigative procedures were straightforward. Techniques and policies focused on computer drives, which investigators inspected in their entirety. Some commercial tools were available to facilitate their work. Investigators were self-taught as formal training had not yet been developed.

A series of changes rendered this early approach ineffective. Rather than being the objects of crimes, computers were used to facilitate crimes. The technology itself changed; computers were able to store massive amounts of data, and some technology did not even exist in the 1990s—cloud computing, for example. Crimes were committed by entities such as nation-states and not just individuals. Criminals were often sophisticated technical experts; for example, individuals with PhDs in computer science "gone rogue." Many attacks were so complex and large in scope that the commercial tools became ineffective. As both dependence on the internet and the sophistication of attackers grew, law enforcement investigators faced many challenges, including a huge backlog of computers waiting for analysis, a growing volume of digital evidence, and attackers' use of encryption. The investigators needed highly technical training and more effective tools and techniques.

## 8.0.1 SEI Entry into Digital Intelligence and Investigation

The SEI became involved in forensics in response to the August 2003 U.S. Northeast blackout and, subsequently, expanded its operational support for investigations into cyber attacks. With the hiring of a former law enforcement professional, the SEI established a separate team, with the former law enforcement agent as the team lead.

The SEI was in a good position to provide support to law enforcement investigators and help them meet their challenges. Through the CERT/CC, SEI researchers had experience with responding to computer security compromises, with vulnerabilities and ways they could be exploited, and with malicious code. Through the malicious code work, in particular, the SEI had well-established, trusted relationships with law enforcement agencies.

## 8.0.2 Evolution of the SEI Approach

The SEI continued to gain operational experience by assisting federal law enforcement agents with their cases. One example is the Iceman case. A former computer security consultant, Max Ray Butler (also known as the Iceman), attacked computers at financial institutions and credit card processing centers, stealing account information and selling the data to others. Federal law enforcement agents enlisted the SEI's assistance in acquiring and decrypting the Iceman's data, thus providing critical evidence for the case that resulted in a three-year sentence for wire fraud and identity theft, plus five years of supervised release and $27.5 million in restitution payments to victims [Mills 2009, McMillan 2010].

In supporting law enforcement cases, the SEI identified gap areas not addressed by commercial tools or commonly used techniques. With the goal of preserving evidence and presenting it in a way that leads to the apprehension of the criminals, the SEI developed tools, analysis methods, and processes that enable comprehensive and efficient analysis of evidence for use in cybercrime cases.

Cases involve large amounts of data and important volatile data. As a first step toward addressing the need to extract and understand the data quickly, the SEI developed the CERT LiveView tool.[63] To deal with the increasing use of strong encryption of data on seized computers, the SEI is developing ways to adapt the data acquisition process and recover encrypted data. SEI-developed technologies, tools, and practices have resulted in previously unattainable results for national and international cybercrime investigations. As a major advance, the Department of Justice influenced the federal government to accept evidence from SEI technology as being admissible in court cases.

By continuing to provide operational support to high-profile intrusion, identity theft, and general computer crime investigations, the SEI is able to see the changing limitations of computer forensics and incident response in the field first-hand. Combining this applied research with the talents, operational experience, research capabilities, and the extensive knowledge base of Carnegie Mellon University, the SEI will remain unmatched in its ability to develop new tools and methods to address cybersecurity limitations and critical gaps.

### 8.0.3 Influence on the State of the Practice

The continuing SEI digital intelligence and investigation advances are used primarily by law enforcement. In addition, the SEI makes some tools available to system administrators through the web.[64] With the SEI tools and techniques, system administrators can identify malicious activity and establish a chain of evidence. As a result, criminals may be stopped before they cause more damage.

To increase the government's capability to deal with attacks, the SEI provides training to federal, state, local, and international law enforcement agencies in the use of its tools and techniques. The staff also teaches an SEI course on forensic response and analysis[65] and presents courses in the Cyber Forensics and Digital Response track at Carnegie Mellon's Information Networking Institute (INI), which offers certification in digital forensics.[66] The CERT STEPfwd training environment includes demonstrations that show how to use some of the SEI-developed digital investigation tools.

### 8.0.4 Keeping Up with Changes in Cybercrime

Now the SEI assists in the pursuit of cybercriminals and develops tools and methods that both prevent and combat cybercrime. Future research and development will enable the SEI to keep up with changing technology, risks, attacks, and federal law enforcement and incident responders' needs. Operational support will help ensure that the SEI is focusing on essential gap areas. The

63 See http://liveview.sourceforge.net

64 See http://www.sei.cmu.edu/digitalintelligence/tools

65 A course description can be found at http://www.sei.cmu.edu/training/P103.cfm

66 For information about INI, and its Pittsburgh programs in particular, see http://www.ini.cmu.edu

ultimate contribution of SEI digital intelligence and investigation work is that it makes cyberspace safer for government, critical infrastructure operators, other industries, and individuals to conduct their essential activities.

### 8.0.5 References

[McMillan 2010] McMillan, Robert. “Criminal Hacker ‘Iceman’ Gets 13 Years.” *Computer-World*, 2012. http://www.computerworld.com/s/article/9156658/Criminal_hacker_Iceman_gets_13_years (2012).

[Mills 2009] Mills, Elinor. “‘Iceman’ Pleads Guilty in Credit Card Theft Case.” *CNET*. 2009. http://news.cnet.com/8301-1009_3-10275442-83.html (2009).

[Nolan 2005a] Nolan, Richard; O’Sullivan, Colin; Branson, Jake; & Waits, Cal. *First Responders Guide to Computer Forensics* (CMU/SEI-2005-HB-001). Software Engineering Institute, Carnegie Mellon University, 2005. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=7251

[Nolan 2005b] Nolan, Richard; Baker, Marie; Branson, Jake; Hammerstein, Josh; Rush, Kristopher; Waits, Cal; & Schweinsberg, Elizabeth. *First Responders Guide to Computer Forensics: Advanced Topics* (CMU/SEI-2005-HB-003). Software Engineering Institute, Carnegie Mellon University, 2005. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=7261

## 8.1 Operational Support for Digital Intelligence and Investigation

### 8.1.1 The Challenge: Catching and Convicting Perpetrators of Cyber Attacks

Attacks on internet-connected systems put government, business, and consumers at risk. The nature of the internet makes their networked systems and the information on them vulnerable to compromise. At first, computers were the objects of attacks. Law enforcement investigators could examine all the data on a hard drive and had commercial tools to assist them. Now computers are used to facilitate crime, and the traditional tools and methods cannot keep up with the velocity of investigations, preventing law enforcement agents from establishing timely leads. The attackers are sophisticated and the attacks are complex. Their targets include the Department of Defense and other federal computers, commercial organizations, educational institutions, and critical U.S. infrastructure. Huge volumes of data are involved, more data than an investigator could examine in its entirety even with the commercially available tools. Complicating the investigation are volatile data and criminals' use of encryption. Law enforcement investigators need tools, techniques, and training that enable them to extract important data, analyze it, and catch the perpetrators. Administrators of attacked systems need to be able to identify intrusions and recover from attacks while preserving the chain of evidence that leads to the attackers' arrest and conviction.

### 8.1.2 A Solution: Tools and Techniques for Digital Investigations

The SEI became involved in forensics in response to the August 2003 U.S. Northeast blackout and, subsequently, expanded its operational support for investigations into cyber attacks. From the start, SEI experts have focused on how to preserve evidence and present it in a way that leads to the apprehension of the criminals, and their approach has included comprehensive and efficient analysis of evidence, supported by technology and tools. The SEI continues to identify gap areas to address while supporting law enforcement operations such as these:

- The SEI assisted an investigation of the theft of more than 90 million credit and debit card numbers from T.J. Maxx, Marshall's, Barnes & Noble, OfficeMax, and other major retailers [Houser 2008]. Referred to as the TJX case, the theft occurred in 2005 and constituted one of the largest instances of credit card fraud and identity theft in history [Moore 2010]. While assisting with law enforcement's analysis, the SEI developed a new tool for recovering and organizing credit card numbers from digital evidence. Two SEI staff members received the U.S. Secret Service Director's Recognition Award for their contributions to the TJX case. U.S. representatives also recognized the team's efforts during a visit to Carnegie Mellon University [FedNews 2008]. Eleven individuals were indicted in 2008 for the data breach, including the leader, Albert Gonzales. The government claimed in its sentencing memo that companies, banks, and insurers lost close to $200 million and that Gonzalez's credit and debit card thefts "victimized a group of people whose population exceeded that of many major cities and some states" [Zetter 2010].
- A similar attack by the same perpetrators occurred in the Heartland case [Barrett 2009]. The SEI assisted the U.S. Secret Service with investigation into intrusions and credit and debit card theft at Heartland, Hannaford Bros., 7-Eleven, and three other retailers. The leader, Albert Gonzales, and two Russian accomplices were indicted in August 2009. Gonzales, who led both major attacks, received two 20-year sentences and was required to make restitution to victims.

The SEI's experience with cases such as these gives its forensics experts first-hand knowledge of the needs of the law enforcement community. They develop tools, analysis techniques, and processes that meet a real need in the field. They also provide guidance to management on policies and practices [Waits 2007].

### 8.1.3 The Consequence: Cyber Attackers Are Caught and Prosecuted

Because of the SEI's involvement, along with the tools and techniques it develops, law enforcement investigations are more efficient and effective. Criminals such as Gonzales are convicted of their crimes, paying retribution and serving time in prison. In a major step forward, the federal government now accepts evidence from SEI technology as being admissible in court cases. In addition, system administrators know how to identify malicious activity and establish a chain of evidence. As a result, criminals may be stopped before they cause more additional damage, and the evidence holds up in court. Criminals are convicted, preventing them from attacking the networks of the DoD, federal government, commercial endeavors, and critical infrastructures.

**The View from Others**

*The SEI continues to be an integral partner in the Secret Service's efforts to combat cybercrime and protect the nation's critical financial infrastructure.*

– Tom Dover, U.S. Secret Service [SEI 2009]

### 8.1.4 The SEI Contribution

The SEI develops tools and analysis techniques (described further in another section) and uses them operationally to assist the Secret Service and other law enforcement agencies to convict perpetrators of cybercrimes. In 2011, the SEI provided advanced analytical support for 147 federal investigations. Depending on the needs of each case, they may support digital investigations by collecting data, addressing encryption and configuration issues on attackers' computers, creating forensic images for analysis, assisting with analysis, and even testifying in court if required. Because the SEI shares knowledge and tools and provides training, both law enforcers and system administrators have greater skills, leading to the conviction of more criminals.

### 8.1.5 References

[Barrett 2009] Barrett, Devlin. "'Soupnazi' Albert Gonzales: Jailed Miami Man Hacked into 130 Million Credit Card Accounts: Prosecutors." http://www.huffingtonpost.com/2009/08/17/prosecutors-jailed-miami_n_261378.html (2009).

[FedNews 2008]. "CMU Software Engineering Institute Recognized by Reps. Murtha, Doyle, and Altmire." *U.S. Fed News* press release 2008.

[Houser 2008] Houser, Mark. "CERT Helped U.S. Crack International ID Theft Case." *Tribune-Review*, 2008. http://triblive.com/x/pittsburghtrib/news/education/s_586552.html#axzz3GPsgTRg8 (2008).

[Moore 2010] Moore, Kevin. "TJX and Heartland: The Role of the CERT Forensics Team" (podcast). Software Engineering Institute, Carnegie Mellon University, 2010. www.cert.org/podcasts/podcast_episode.cfm?episodeid=34383 http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=30936

[SEI 2009] *SEI Year in Review*. Software Engineering Institute, Carnegie Mellon University, 2009. http://resources.sei.cmu.edu/asset_files/AnnualReport/2010_001_001_30135.pdf

[Waits 2007] Waits, Cal. "Computer Forensics for Business Leaders: Robust Policies and Practices" (podcast). Software Engineering Institute, Carnegie Mellon University, 2007. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=34414

[Zetter 2010] Zetter, Kim. "TJX Hacker Gets 20 Years in Prison." *Wired*. March 25, 2010. http://www.wired.com/threatlevel/2010/03/tjx-sentencing (2010).

## 8.2 Digital Intelligence and Investigation Methods and Tools

### 8.2.1 The Challenge: Effective and Efficient Cyber Forensics

Traditional approaches are insufficient for investigating crimes committed through computers. Early digital forensic tools and techniques were helpful when the amount of data stored on computers could be examined fully; however, they became insufficient when the scope, frequency, and complexity of cybercrimes significantly increased. As a result, federal law enforcement agencies soon faced a backlog of digital evidence to analyze and a volume of data that could not be examined effectively with the commercial tools available. The attacks they now investigate have become more complex and are perpetrated by sophisticated attackers, who often use encryption and other means to obscure their activities. Additionally, system administrators need to understand attacks on their systems and preserve evidence in a way that holds up in court.

### 8.2.2 A Solution: Tools and Methods that Improve the State of the Practice

The SEI forensics experts develop highly specialized computer forensics and incident response "gap area" tools and practices not addressed by commercial tools or standard techniques (see the report by Waits [Waits 2008], for example) and provide them to the DoD and U.S. federal civilian law enforcement agencies. Thus, the SEI equips federal law enforcement investigators to efficiently and effectively handle cyber attacks, from collecting evidence to apprehending and convicting the perpetrators. Because SEI forensics experts work on incidents involving national security and assist with large-scale criminal cases, they gain the essential field experience that helps them identify the areas that need to be addressed, and their close working relationship with law enforcement gives them the insight they need to focus their efforts.

Because the scale of incidents and the amount of data prevents investigators from examining all data, the SEI has identified triage strategies and automated tools for computer and data acquisition that result in actionable information and evidence that has proven to hold up in court. Because a criminal can commit a crime and disappear before traditional response approaches can be implemented, the SEI identifies techniques for rapid response without jeopardizing sensitive data. The SEI developed its first forensics support tool, Aperio, in 2004-2005. By 2012, six tools were freely available to system administrators, with an additional four provided to law enforcement only.[67] In addition, the SEI established and maintains a repository of Linux forensics tools[68] that are packaged for easy download and installation by any practitioner who must acquire and analyze data. Other SEI advances include rapid triage and correlation of malicious code and network logs/traffic; new technologies to improve DoD investigators' collection of intelligence from recovered media; new methods and techniques for forensic imaging of solid state drives; and tools for image and video analysis.

The SEI developed the CERT Clustered-Computing Analysis Platform (C-CAP), a technological advance to address the need for forensics law enforcement analysts to work together on cases, even if they are geographically dispersed. SEI developers integrated access to a comprehensive array of analytical tools and resources. C-CAP is centrally managed, so platform resources can be

67 Tool descriptions can be found at http://www.sei.cmu.edu/digitalintelligence/tools

68 See http://www.cert.org/digital-intelligence/tools/linux-forensics-tools-repository.cfm

allocated rapidly and can be flexibly and securely reassigned on demand. The resources are scalable, so that functionality, storage, and processing power can meet changing demands.

The SEI has developed training to help practitioners use its tools, techniques, and processes effectively in the field, further increasing the government's ability to apprehend and convict criminals. In addition, the SEI staff worked with Carnegie Mellon University's Information Networking Institute (INI) to define and teach in a master's-level curriculum in digital forensics [Rush 2010], which is described on the INI website.[69]

### 8.2.3 The Consequence: Cyber Criminals Are Caught Early and Prosecuted

As a result of SEI forensics research, gap area tool development, and training, the DoD and federal agencies have an increased capability to deal with cyber attacks. Law enforcement investigators can effectively collect and analyze evidence that helps them catch and convict criminals; the forensics investigators and analysts have an enhanced ability to rapidly identify the method, source, and impact of cyber attacks.

Supporting their cases are system administrators who know how to preserve the chain of evidence as they use SEI tools to understand malicious activity on their systems and collect evidence that is usable in court. By identifying malicious activity and establishing a chain of evidence, system and network administrators may stop criminals before they cause additional damage. They use SEI tools to examine their systems following an intrusion even while attackers continue to find more sophisticated ways to hide their actions. They have a greater ability to preserve evidence while restoring their systems.

Through C-CAP, analysts and investigators enjoy flexible, secure access to high-performance systems, increasing productivity and enabling collaboration.

A significant consequence of SEI work is that the Department of Justice has approved several SEI technologies as being admissible in court cases. As a side effect, the acceptance of evidence from SEI tools has led to an increased demand for the tools and for C-CAP, thus further improving the forensic capacity of government law enforcement agencies. In 2013, there were more than 20,300 downloads of SEI-developed forensics analysis tools. In addition, there has been a total of 200,000 individual downloads of LiveView alone since it was released as the first tool in 2006. The ultimate consequence of SEI research and development is the imprisonment of cybercriminals, affecting national security, and a safer internet environment for DoD, federal agencies, and U.S. businesses.

### 8.2.4 The SEI Contribution

The SEI is filling gaps in techniques and tools not addressed by traditional forensics techniques and the commercial tools that support law enforcement analysts. Because of the close relationship with law enforcement, the SEI can identify areas that directly address the agencies' needs. The SEI focuses on core issues facing the law enforcement agencies, such as the need to process large amounts of data quickly. By concentrating on the core issues, the SEI is able to create solutions

69 See http://www.ini.cmu.edu/degrees/

that not only apply to the specific cases but can be amplified for broader use. To explore alternative solutions, SEI experts have collaborated with researchers in Carnegie Mellon's School of Computer Science, Robotics Institute, Department of Electrical and Computer Engineering, and the CyLab Biometrics Center.

The major SEI contribution is the increased ability to convict cybercriminals. The three largest cases prosecuted by the U.S. Department of Justice were direct results of SEI technology that supports law enforcement analysts.

### 8.2.5 References

[Rush 2010] Rush, Kristopher. "Computer and Network Forensics: A Master's Level Curriculum" (podcast). Software Engineering Institute, Carnegie Mellon University, 2010. http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=34391

[Waits 2008] Waits, Cal; Akinyele, Joseph; Nolan, Richard; & Rogers, Lawrence. *Computer Forensics: Results of Live Response Inquiry vs. Memory Image Analysis* (CMU/SEI-2008-TN-017). Software Engineering Institute, Carnegie Mellon University, 2008. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=8605

# 9 The Future of Software Engineering

## 9.1 A Vision of the Future of Software Engineering

### 9.1.1 Software and Defense

In the years since the SEI was founded, the Department of Defense has become dramatically more reliant on software in all phases of the DoD mission. Software is a critical building material for major systems of all kinds. Considering the pervasiveness of its role, it is perhaps the most strategically significant of all materials that contribute to modern defense systems.

The increase in the role and criticality of software is well documented in multiple studies by the Defense Science Board (DSB) and the National Research Council (NRC) at the National Academies.[70] This increased dependency is evident in the reported tenfold growth in the number of lines of software code produced every decade. There are good reasons for this growth: software is uniquely unbounded and flexible; can be delivered and upgraded electronically and remotely; and has the potential for rapid adaptation to changing threats, coalition structure and mission environment, and technological infrastructure. That is, software, uniquely, enables defense program managers and sustainment teams to adapt rapidly to changes in their missions, as well as to opportunities afforded by rapid advances in the computing and communication infrastructure. In addition to emerging as the most compelling medium for embodying complex functionality, software has become the critical medium of interoperation and interconnection among systems. These advances in software capability have a broad and deep significance throughout the DoD and its supply chain.

Along with the rapid growth in the role of software in DoD systems, there is rapid evolution in the technology and practices associated with software, which is increasing rather than diminishing. It is dangerous to think that software technology is somehow approaching a plateau in role and capability. The lack of a plateau is evident in recent developments in technology and practice—examples include big data frameworks, machine learning tools, advanced framework-and-apps architectures, safe programming languages, cyber-physical architectures, resilient architectures, software-assurance analytics, design modeling, cost estimation, agile practices, and the like. As software-related technology evolves, organizations and nations that are deeply reliant on software capability must take an active role in engaging with the technology and the technology ecosystem, lest they fall behind their competitors and adversaries.

70 Principal related studies from the National Research Council include *Critical Code: Software Producibility for Defense* (2010), *Achieving Effective Acquisition of Information Technology in the Department of Defense* (2010), *Software for Dependable Systems: Sufficient Evidence* (2007), and *Toward a Safer and More Secure Cyberspace* (2007). Principal related reports of Defense Science Board Task Forces include *Mission Impact of Foreign Influence on DoD Software* (September 2007), *Defense Software* (November 2000), *Resilient Military Systems and the Advanced Cyber Threat* (January 2013), *Acquiring Defense Software Commercially* (June 1994), and *Military Software* (September 1987).

### 9.1.2 The SEI Role

The SEI works with diverse organizations within the DoD—and also in other agencies and in the DoD supply chain—to leverage the rapid changes in software technology to support diverse facets of the mission at all stages of the lifecycle, including development, assurance, and sustainment for cyber-enabled missions. In its FFRDC role, the SEI develops methods to manage risk, cost, schedule, quality, security, and complexity of software-reliant systems. The SEI works throughout the software supply chain, including with prime government contractors, their subcontractors, supporting vendors, open-source foundations, government labs, other FFRDCs, and researchers. Through these relationships, the SEI learns how the DoD can both make the best use of and also facilitate the advance of DoD-essential technologies that are emerging in the commercial sector, as well as the new technologies that are still being evaluated in the lab.

A key challenge in the shifting software landscape is that advances in software technology and practice consist of much more than incremental enhancements to established existing practices and tools. We must also focus on advancing and redefining the practice to support new techniques that are better suited to delivering the capability, flexibility, and assurance necessary for modern systems. We call this *software producibility*—the capacity to design, produce, assure, and evolve innovative software-intensive systems in a predictable manner, while effectively managing risk, cost, schedule, and complexity.

### 9.1.3 Looking Ahead

In this section, we look forward in software engineering, in related topics in cybersecurity, and in potential roles for the SEI as an institution. There are significant changes underway in the ways we develop, assure, and sustain software-reliant systems. There are also significant changes in the technologies and practices associated with software, including languages, tools, models, and analyses, as well as supporting hardware and communications infrastructure. This visionary exercise reflects an activity that is ongoing at the SEI, which is to continually look forward with respect to both the mission context and the advancement of technology and practice. This forward-looking process involves direct engagement with stakeholders across a spectrum from basic researchers in universities to mission-focused operators and sustainment managers. The process involves identifying and assessing potentially important points of technical advancement that can provide meaningful leverage on a wide range of challenges associated with developing and sustaining software-reliant systems.

In the past, an active forward-looking research and planning effort has led to the many advancements documented throughout this volume, such as CMM/CMMI, TSP, the scalable conops of the CERT/CC,[71] real-time scheduling theory and practice, and the emergence of architecture practice.

71 W. L. Scherlis, S. L. Squires, R. D. Pethia, “Computer Emergency Response,” in P. Denning (ed.) *Computers Under Attack: Intruders, Worms, and Viruses*, Addison-Wesley, 1990.

In each case, impact is achieved though the attainment of a balance between the ambition and extent of leverage of the new advancement, on the one hand, and the feasibility and transition of the advancement, on the other.

Looking forward, we envision advances in several areas that are critical to the development and sustainment processes, to achieving assured and secure systems, and to creating certain functionality essential to a wide range of software-reliant DoD systems. All these advances occur in the context of significant changes in the structure of major defense systems and the role of software in those systems. For example, software is the key to achieving assured system-of-systems interoperability, integration, and configurations; it enables the shift from platform-centric stovepipe solutions to payload-centric "framework-and-apps" approaches.[72] An emerging theme is the development of robust and scalable architectures that enable component assembly and incremental advancement. This is an important advance for the engineering of large and complex systems, but it also creates significant challenges for software assurance.[73]

In addition, software has become the *materiel* of both defensive cybersecurity structures and offensive cyber capability. Improving our ability to create, manipulate, and analyze software is essential to advancing all aspects of cyber warfare. Finally, supply chains for software are becoming more complex, more diverse, and more international. This evolution is a consequence of recent advances in technology and in computing infrastructure. An unavoidable consequence of rich supply chains is that we must consider attack surfaces not just at the periphery of complex systems, but also within the systems, including the role that humans play as operators and users in these systems.

Below we highlight the four principal dimensions of the SEI technical vision. Each of these encompasses a wide range of technical problems. But each also demonstrates that incremental progress on technical sub-problems can lead to incremental improvements in mission capability.

**1. Architecture-Led Incremental Iterative Development (ALIID).** The goal of ALIID is to enable iterative and incremental development of highly capable and innovative systems with acceptable levels of programmatic risk. While small-team agile methods are well established in most sectors, the challenges of scaling up to larger efforts remain profound. ALIID is intended to manifest the aspiration of "agile at scale" by enabling larger scale iterative and incremental development in DoD development and sustainment/modernization projects.

72 This concept of shifting emphasis in the development of DoD systems from "platform" to "payload" is analogous to the emergence of socio-technical ecosystems for mobile devices (IOS and Android frameworks and apps), for big data analytics (MapReduce frameworks), and in other applications. ADM Greenert articulated the case in an article, Payloads over Platforms: Charting a New Course, in *US Naval Institute Proceedings Magazine*, 2012.

73 This issue is addressed in the report *Mission Impact of Foreign Influence on DoD Software*, Defense Science Board 2007.

Diverse experience in industry suggests that this type of development is almost always accomplished on the basis of early commitment to effective architectural structures. There are two principal reasons for this. First, architectural structures have enormous influence on quality and security outcomes for systems. Without early consideration of quality and security attributes, it may be infeasible to "bolt them on" later in a process. Indeed, for many categories of systems, early architectural decisions can be a greater influence on success than nearly any other factor. Second, architectural structures strongly influence the work breakdown structure for major software-intensive systems. The ability to achieve an effective granularity of effort—breaking larger tasks into feasible subtasks that can be managed separately and at differing tempos—is an architecture-based outcome at nearly every level of scale.

It is easy to see the benefits of the "framework-and-apps" model ubiquitously adopted in the commercial sector for mobile devices, web applications, GUI development, and other categories of systems. This model is a compelling example of architecture-enabled scale-up. At the SEI, we aspire to that same approach in defense software: for example, in the development of principles of design for systems of systems. Military leaders have described a shift from platform-focused approaches to payload-focused approaches. This shift—which is premised on improving agility while lowering costs and risks—is enabled by appropriate architectural concepts.

While many areas of technical development are needed to aggressively advance this vision, it is nonetheless possible in many domains to advance development approaches based on these concepts. The evidence of this feasibility is the widespread adoption of architecture-led approaches in the commercial sector for products, software as a service (SAAS) services, and the like. Indeed, many commercial and in-house government development organizations already employ practices that are increasingly consistent with the ALIID vision. It is a major challenge, however, to adapt these ideas to the kind of arm's length engagement with development organizations characteristic of DoD major defense acquisition programs.

Advancing the ALLIID vision involves at least four areas of particular emphasis:

a. **Architectural structures and practices**. This area includes identification of concepts of operations for systems of systems, ultra-large-scale systems,[74] and the like. This area also includes the development of practices for early validation, to assist in getting reliable early assessment of quality and security outcomes associated with particular architectural choices.

b. **Measurement and process models for cost, progress, and engineering risks**. Any approach based on incremental and/or iterative development must incorporate effective practices for estimating costs, progress, and risks and then use these practices to reward developers for value earned. Cost-estimation techniques, for example, could be enhanced to evaluate both estimated mean values and variances and to support ongoing

74 See https://www.sei.cmu.edu/uls/ for more information.

re-estimation as a means to assess progress. Risk-reduction practices such as prototyping or modeling and simulation may lead, for example, to both reduced "cost to complete" and also narrowed variances (i.e., greater confidence that the estimates are accurate).

c. **Incentives and acquisition practices**. How can acquisition practices be developed to build on advances in measurement and process models that feature a structuring of incentives that enables government and contractors to collaborate effectively in development of architectures and in iteration at scale?

d. **Software sustainment and modernization**. The reality of many sustainment efforts is that they are really supporting a continual evolution and modernization of systems. Planning for continuous evolution, for example in the form of identifying and separating dimensions of variability, is a necessary feature of architectural design. Continuous evolution, importantly, can also involve discontinuous change as infrastructures and subsystems evolve and new choices emerge.

**2. Evidence-Based Software Assurance and Certification.** The goal of this second element of vision is a dramatic reduction in the cost and difficulty of making assurance judgments related to quality and security attributes. Achieving this goal is particularly important as systems become more complex and evolve more rapidly. Current approaches for certification and accreditation are largely based on an after-the-fact evaluation of a snapshot of a system.

While after-the-fact approaches are effective for certain well-defined categories of components and systems, they tend to break down as systems increase in complexity, scale, and dynamism. They also tend to hinder ongoing evolution, rapid reconfiguration, dynamic loading of components, autonomy, and composition and interlinking of systems of systems. Put simply, these established techniques do not scale up, and they do not work well for the emerging software framework-based systems now prevalent in commercial and infrastructural applications.

The industry folklore has long asserted that quality-related activities, including security-related assurance, can consume half of total development costs for larger systems. For example, the *IBM Systems Journal* states that in a typical commercial development organization, "the cost of providing [the assurance that the program will perform satisfactorily in terms of its functional and nonfunctional specifications within the expected deployment environments] via appropriate debugging, testing, and verification activities can easily range from 50 to 75 percent of the total development cost." [Hailpern 2002]. Additionally, after-the-fact evaluation practices can add a year or more to the elapsed time required to develop and deploy software-reliant systems. Commercial systems, including products and software as a service and cloud-based systems, tend to undergo a relatively rapid and continual evolution. For many of our DoD and infrastructural systems, we similarly need to support a continuous evolution.

Some areas of particular technical emphasis include

a. Architecture and composition principles. These enable separate evaluation of individual components, with the possibility of combining results to achieve aggregate assurance

judgments. These principles are motivated by the reality of modern software supply chains, which are rich and diverse in sourcing and geography.

b. Modeling and analytics. There is an extraordinary diversity of quality attributes significant to DoD and infrastructural systems, and each has its own technical approaches to support assurance judgments. Examples include testing, inspection, static analysis, verification, model checking, monitoring, and encapsulation.

c. Tools and practices. Modern software development practices are data-intensive, and there are new good opportunities to incorporate the creation of evidence in support of assurance claims into the process of development. This evidence-based assurance can harmonize incentives to create designs and implementations that can more readily support evaluation.

d. Evaluation and other techniques to support the use of more opaque components in systems. This includes the challenge of undertaking acceptance evaluation for binary components and potentially dangerous components from unknown sources.

**3. Critical Component Capabilities**. The goal of this third element is to enhance DoD software capability in several areas that have critical and pervasive roles in DoD software-reliant systems. These areas include the following:

a. **Composable cyber-physical systems (CPS)**. "Cyber-physical" refers to the fact that embedded software operates in the context of physical sensors and affectors. CPS thus includes control systems, real-time systems, mobile systems, distributed systems, and many other categories of systems pervasive in the DoD and critical infrastructure. These systems tend to benefit from a conventionalization of architectural models and design approaches (such as rate monotonic analysis). But they also may have greater complexity due, for example, to higher coupling and the need to model and manage associated physical system components. They also frequently need to assure that real-time deadlines are met. A consequence is that they typically manifest greater internal coupling in their design, thwarting higher levels of capability, composition, and flexibility.

b. **Networks, networking, and mobile applications.** Network structure and characteristics have a significant influence on software and systems architectures for military systems. Networks at the "edge," such as in-theater, can be subject to greater challenges with respect to both internal and reach-back connectivity. This influences network architecture, security planning, and also the management of data and computation resources critical to warfighting. In-theater networks also provide infrastructure for mobile devices and applications. These applications have particular needs regarding security, context-awareness, usability, and scalability.

c. **Autonomous systems**. Autonomous systems are cyber-physical systems that can accept sensor data and mission guidance, and, with very limited (or no) human interaction, arrive at mission decisions and enact outcomes. These systems are increasingly critical to the defense mission, and yet they pose particular challenges for verification and validation since they rely so much less on ongoing human interaction. Indeed, the capability

and complexity of these systems are often limited for this reason. Systems must both behave as expected and, additionally, not manifest unwanted behaviors that can be dangerous or threaten mission success. From a technical perspective, autonomy can be particularly challenging because of the vast state space, the number of possibilities of combinations of inputs, challenges of error tolerance, and the difficulty of fully modeling the environmental circumstances of their operation.

d. **Data-intensive systems for analytics and for modeling and simulation**. Advances in sensor fidelity, rapid growth in network capacity, increasing convergence in data center and high-performance computing architectures, advances in large-scale data storage, and emerging frameworks for scalable distributed computing (such as MapReduce and GraphLab) have all resulted in the growing phenomenon of "big data." There are many significant applications of big-data techniques in DoD and infrastructural systems—and in the development of those systems as well. Indeed, many of the other features of the SEI Strategic Research Plan build on progress in big data.

**4. Cybersecurity Tradecraft and Analytics.** The goal of the fourth strategic element is to advance analytic capability in support of diverse aspects of the cybersecurity mission. These aspects include analytics and situational awareness for malware; vulnerability categorization and assessment; vulnerability information management; network activity analysis; threat characterization and assessment, including insider and supply-chain threats; organizational security; and many other dimensions of operational response, remediation, and recovery. In addition, there is increasing emphasis on the theoretical underpinnings in cybersecurity. Advances in foundational work can assist in the formulation of technical and operational strategies that go beyond purely reactive approaches.

This capability builds on a range of data assets and tooling related to adversarial tradecraft, malware, vulnerabilities, insider threats, and other results of experience with large numbers of cybersecurity-related incidents. There are diverse purposes of this strategic element, including

a. Improvement in our understanding and communication of threats and risks and adversarial intent
b. Development of better preventive approaches in the engineering of systems and in managing secure operations, including considerations for security and assurance "at scale," improved indications and warning, and near-real-time data analysis
c. Support for forensic and corpus analysis
d. Support for hypothesis generation using machine learning, near-real-time analysis, and other advanced capabilities

The future of software and cybersecurity will go well beyond incremental improvements on the current baseline of capability, quality, security, and productivity. The four areas outlined above represent features of an emerging vision of this future capability. This vision encompasses development and sustainment processes, means to achieve more highly assured and secure systems, and mechanisms associated with creating functionalities most essential to a wide range of DoD sys-

tems. The SEI is implementing collaborative strategies to make progress in these technically challenging areas, and we expect that this activity will make significant differences in the practice of software and cybersecurity.

### 9.1.4 Reference

[Hailpern 2002] Hailpern, B. & Santhanam, P. “Software Debugging, Testing, and Verification.” *IBM Systems Journal 41*:1 (2002).

# 10 Conclusion

## 10.1 Leading the Community

Charged with a challenging mission, the SEI has responded by building a mature organization that has not only kept abreast of evolving technology, but also has consistently anticipated DoD needs and prepared technology solutions to meet those needs. At the same time, the SEI has provided leadership to the broader software engineering community.

A major contributing factor in the SEI's leadership position has been the early strategic decision to engage the community in its efforts to achieve the greatest impact. In essence, the SEI identifies a problem area important to the DoD and the broader software engineering community, recruits a recognized leader in that area, and then engages the community in a consensus-building effort to produce a solution. The community responds because it respects the SEI's independent position, free of commercial or government bias. As the SEI's reputation has grown, leaders in the community have become willing to cooperate because they want to influence the direction and they respect the SEI's proven ability to lead them to a consensus. It is not uncommon for senior corporate officers at the level of director of engineering or even vice president, senior research fellows, senior government officials, and leading academics to participate in SEI-led efforts. International participation is also the norm. These external contributions are often significant, involving many hours of collaboration; and the results are beyond what any organization could produce on its own. More importantly, the results are accepted by the community.

Much of the work described in this volume is a result of such a process, refined over time. The following are a few notable examples:

- the master of software engineering curriculum and the undergraduate curriculum in software engineering
- the *Ada Adoption Handbook*, which provided guidance to DoD program managers in the adoption of Ada
- the Capability Maturity Model for Software and its derivatives
- the CERT Coordination Center (originally the Computer Emergency Response Team/Coordination Center) processes
- the International Process Research Consortium's outline of process research
- the ultra large scale systems definition of software engineering research directions
- the Architecture Analysis and Design Language (AADL), which has been adopted as an international standard for avionics systems
- the OCTAVE method's use for compliance with the Health Insurance Portability and Accountability Act (HIPAA)

## 10.2 Highlighting 30 Years of Contributing to DoD Software Capability

As the preceding pages have detailed, the SEI has lead the adoption of significant improvements that have changed the nature of software engineering.

### 10.2.1 Real-Time Embedded and Cyber-Physical Systems

Thirty years ago, software for real-time embedded systems was developed largely in assembly language, with few supporting tools and with software architectures that were often inappropriate for the task and schedulers that were developed using ad-hoc analysis. Today, software engineers have architecture models for real-time systems and analytic techniques for designing schedulers that will prevent failure; they confidently construct such systems in high-level languages.

Specifically, the SEI

- assisted the DoD with several technical aspects of Ada adoption and provided "honest broker" guidance to the software development community
- developed a real-time testbed for assessing the quality of compilers and runtime systems
- with CMU faculty, developed rate monotonic analysis, which provided the first engineering basis for developing real-time schedulers
- extended the analysis method to multi-core processors
- developed an architecture that allows for safe operation when the system is composed of a safe component and a less reliable component
- led the development of an Architecture Analysis and Design Language (AADL), which is an international standard and used on a variety of DoD systems, particularly guidance systems

### 10.2.2 Software Engineering Education and Training

Thirty years ago, there was no accepted curriculum for software engineering and few universities were teaching software engineering-related courses. Today, nearly all university software engineering-related curricula trace their lineage to SEI-led efforts, including undergraduate and master's degree software engineering curricula, software assurance curricula, and survivability and information assurance curricula for system administrators.

Specifically, the SEI

- led the development of a master of software engineering (MSE) curriculum that is used by most universities
- offered video courses of its joint MSE program to a large audience, thereby accelerating the adoption of the curriculum
- led the development of an undergraduate curriculum from which most universities tailor their programs
- established a mechanism for managing future curriculum development through the IEEE
- trained more than 60,000 people involving 65 different courses
- established a partner network of companies authorized to provide SEI-developed training

- established an online training program for network security professionals and for forensics analysts

### 10.2.3 Management

Thirty years ago, managing the software development process for even a 100,000 lines-of-code system was widely acknowledged as chaotic, leading to extensive delays, overruns, and catastrophic failure. Today, organizations have models for the management process, which they are able to continuously monitor and improve to successfully manage the development of systems with multi-million lines of code, with predictable cost and schedules.

Specifically, the SEI

- led the development of an internationally adopted capability maturity model for software that enabled disciplined management of the software engineering process
- developed a mechanism for DoD acquisition to evaluate the software engineering maturity of organizations proposing work for the DoD
- developed a team approach to software engineering that enabled teams to perform in a disciplined manner and achieve significant improvements in productivity and error rates
- led the development of a software risk management system for acquisition programs
- developed several capability maturity systems for specialized application, including personnel, systems, and resiliency, that have been adopted widely

### 10.2.4 Security

Thirty years ago, engineers paid little attention to security when building most software and most networks. Indeed ARPANET, which was the predecessor of the internet, was operated with no security under the assumption that there was no threat. Today, security is an integral part of software design, and an active community supports computer and internet security to counter the growing threat of fraud, theft, and espionage.

Specifically, the SEI

- initiated the Computer Emergency Response Team/Coordination Center, now the CERT Coordination Center (CERT/CC), building relationships with numerous companies and academic and government experts to collectively react to network security incidents. During its first 20 years of operation, the CERT/CC processed more than 3 million messages, 25,000 hotline calls, and 300,000 incident reports; cataloged nearly 45,000 vulnerabilities; and published over 2,500 vulnerability notices
- developed a malicious code database and analysis capability and not only maintained an Artifact Catalog but also provided tools and training to the community
- developed the FIRST (Forum of Incident Response and Security Teams) network—an international collection of thousands of organizations that collaborate on network security incidents
- developed secure coding guidelines and standards to help developers prevent vulnerabilities

- developed network situational awareness tools and analysis techniques for quantitatively characterizing threats and targeted intruder activity
- developed cybersecurity engineering solutions to incorporate security early in the software development cycle

### 10.2.5 Engineering Methods

Thirty years ago, there was scant engineering basis to support large-scale software development. As a consequence, problems abounded—including a lack of a theoretical basis for software configuration management, poor understanding of an open systems approach, scarce information on software technologies, and the absence of an integrated, tool-based software engineering environment. Today, engineering tools, methods, and practices underpin the development of software-intensive systems throughout the lifecycle.

Specifically, the SEI

- devised software configuration management concepts and practices on which the software community could base the development of tools, methods, and practices
- developed and codified a body of expertise on computer-aided software engineering (CASE) practices, resulting in several commercially available CASE products
- led efforts to inform and educate the community about open systems practice[75]
- created the Senior Technical Review Group, providing the Air Force and other organizations with a reliable source for information on technologies
- developed methods for re-engineering, such as Options Analysis for Reengineering (OAR), which helps programs to more effectively identify and mine legacy software components
- led an international community in developing key concepts, methods, and practices for managing and engineering systems built using commercial off-the-shelf (COTS) products, including several education courses and the Evolutionary Process for Integrating CBS (EPIC)[76]
- enabled customers to realize the benefits of net centricity through the development of expertise in service-oriented architecture (SOA)

### 10.2.6 Software Architecture

Thirty years ago, architecture was a hardware concept, and few even used the term in connection with software. Today, software architecture is an important consideration in the acquisition and

75 This effort included work with the DoD Open Systems Joint Task Force (OS-JTF), the creator of MOSA (Modular Open Systems Approach), and education, not only of the U.S. defense community (including GAO), but also of allied defense communities in the United Kingdom, Canada, Australia, and New Zealand. It also included the leadership of an IEEE Portable Operating System Interface (POSIX) working group for real-time POSIX specifications.

76 IBM Rational licensed EPIC and used it as the basis for a CBS (computer-based system) plug-in for the Rational Method Composer 7.5 process tool.

design of software-intensive systems; most universities even offer a course in software architecture.

Specifically, the SEI

- was active in addressing software architecture as a discipline before it became widely recognized as a software engineering topic
- developed reference architectures for specific applications, such as flight simulators and user interfaces
- developed a feature-oriented domain analysis technique that served as the basis for a plethora of domain analysis tools available today
- developed methods to give engineers a methodology for identifying and analyzing important software architecture decisions
- focused on the importance of quality attributes as an architectural driver and developed a method for addressing non-functional quality attributes as part of the software architecture
- produced guidelines for developing a software product line and supporting practices that have been used successfully by the DoD and industry organizations to gain significant improvements in their ability to evolve software across multiple platforms

### 10.2.7 Computer Forensics

Thirty years ago, the word *forensics* was not used in conjunction with computers, and the notion that one could extract forensic information from a computer was not considered outside the research community. Today, law enforcement is able to conduct sophisticated forensic analysis on computers used in criminal activity, learn new techniques in a virtual environment, and collaborate with one another in that virtual environment.

Specifically, the SEI

- used its expertise in vulnerability analysis and network intrusion detection to provide support to computer forensics analysts in several government agencies
- developed techniques and supporting tools and training to help computer forensics analysts address high-profile intrusions and identity theft, and investigate computer crime
- provided training for government computer forensics analysts to enable them to stay current with the latest tools and methods used by cybercriminals
- developed forensic technology from which evidence is admissible in federal court computer crime cases
- developed a state-of-the art environment that enables geographically dispersed analysts to access to tools and computing resources and, thus, to cooperate on cases
- defined a master's-level curriculum in digital forensics that has been implemented at Carnegie Mellon University's Information Networking Institute

## 10.3 Direct Support to Government Systems Developers

The SEI has been at the forefront of most software engineering technology developments over the past 30 years and has established itself as a leader to whom the defense software engineering community turns for insight and solutions. The SEI is not a consulting organization in the traditional sense. Nevertheless, it has leveraged its expertise by providing direct support to many DoD and other government organizations facing unique challenges.

The SEI regularly receives comments from government officials thanking it for its contribution to their efforts. These differ from the kinds of acknowledgement received for specific technologies outlined in the technology subsections in that they normally refer to specific individuals and normally cover a range of support that the SEI provided. Since the following are simply examples, the names of the SEI individuals, when mentioned, are replaced by (…).

*I am writing to commend the CMU/SEI handbook called QUASAR and its authors. The F-35 Joint Strike Fighter (JSF) Program used it to assess the computer system architectures of our aircraft and ground systems. It helped us immensely in focusing attention on often-neglected quality attributes, rather than solely upon functional or component-based views of those systems. It guided us in both technical and managerial approaches to architecture assessment. QUASAR enabled the F-35 Program to verify fulfillment of its contractual architectural requirements, and in so doing, improve the quality of the product. QUASAR's basis in CMU/SEI's real-world assessment experience, including on the F-35, undergirds its credibility and veracity. During the past four and one half years, F-35 used QUASAR to successfully assess major subsystems on nine occasions. I participated in the planning or execution of all these events, in my capacity as Mission Systems Architect, and later as Air System Architect. The handbook helped coordinate the efforts of the assessment teams (comprising the Program Office plus CMU/SEI and other subject matter experts) with system designers (comprising the air system contractor—Lockheed Martin, plus its suppliers). I heartily recommend the continued use and development of this valuable tool.*

— Mike Bossert, Mission Systems Architect, JSF JPO (Joint Strike Fighter Joint Program Office) Mission Systems [Firesmith 2010]

*Continued SEI support to Global Hawk Program Office is vital to the continuity of classified program interdependencies and interactions.… SEI provides technical continuity in support of OSD and AF leadership directives with respect to AF UAS Command and Control Initiative standards (UCI) and also holds technical leadership positions in OSD's UAS Control Segment Working Group (UCS). …Additionally, SEI provides significant software engineering support to two Ground Segment efforts valued at over ~$100M—domain expertise and assistance to contractor audits and technical reviews are essential to Global Hawk's success.*

— Carlin Heimann, Col, U.S. Air Force, Global Hawk System Program Director AFLCMC/WIG

### 10.3.1 References

[Firesmith 2010] Firesmith, Donald. *Quality Assessment of System Architectures and their Requirements (QASAR)*. Software Engineering Institute, Carnegie Mellon University, 2010. http://www.sei.cmu.edu/library/assets/presentations/SoSECIE-webinar-2010-firesmith.pdf

## 10.4 An Experienced Staff Well Positioned to Continue Leadership

The SEI is fortunate to have a staff of recognized leaders in software engineering. Many established those reputations in industry, government, or academia before joining the SEI. Others have established themselves through their SEI activities.

In addition to academic degrees in disciplines related to software engineering, the SEI has people who have been recognized by their peers in a variety of ways:

- recipient of the National Medal of Technology
- member of the National Academy of Engineering
- fellows of the IEEE, ACM, and the American Institute of Aeronautics and Astronautics
- distinguished members of the ACM
- recipient of the USENIX Lifetime Achievement award
- conference chairs, program chairs, members of program committees
- invited keynote speakers, invited talks

Another form of recognition is the significant number of SEI publications that have been referenced by others, demonstrating SEI technical leadership.

## 10.5 External Evaluations by DoD Sponsor

As part of the normal due-diligence in advance of contract renewal, the sponsoring agent has conducted a comprehensive review of the SEI involving a blue-ribbon panel of government, industry and academic experts, including representation from government customers and industry partners. The primary purpose of those reviews is to confirm the continuing need for the SEI and its efficacy in executing its mission by conducting the following:

1. an examination of the sponsor's special technical needs and mission requirements that are performed by the FFRDC to determine if and at what level they continue to exist
2. consideration of alternative sources to meet the sponsor's needs
3. an assessment of the efficiency and effectiveness of the FFRDC in meeting the sponsor's needs, including the FFRDC's ability to maintain its objectivity, independence, quick response capability, currency in its field(s) of expertise, and familiarity with the needs of its sponsor
4. an assessment of the adequacy of the FFRDC management in ensuring a cost-effective operation [DoD 2014]

In the course of these reviews, the panel interviews customers, reviews documents and solicits inputs from a variety of sources in and outside the government. In answering the three questions the panel makes overall assessments and some specific recommendations for change both with the SEI program, changes to the sponsoring agreement and changes with government oversight.

The following are excerpts from the Executive Summary of these reports over the last 30 years:

> *The SEI has proven to be a leader in software engineering—a discipline essentially non-existent ten years ago. The SEI's software process products, such as the Capability Maturity Model and its organizational process appraisal methods, have been especially useful to the Government and Industry. The widespread adoption of these products demonstrates the SEI's ability to mobilize the software engineering software improvement.*
>
> —Blue-Ribbon Panel Comprehensive Review of the Software Engineering Institute [DoD 1994]

> *The SEI continues to maintain a world-class reputation as a software engineering research and development center. The SEI provides technical leadership to advance the practice of software engineering so that DoD can acquire and sustains its software-intensive systems with predictable and improved cost, schedule and quality*
>
> —Comprehensive Technical Review of Software Engineering Institute Programs [DoD 2004]

> *The SEI is an independent, unbiased, agent well-suited to assess the current state of any given approach, technology, or process and provide general guidelines and recommendations to its sponsors. …The SEI is uniquely capable. Staff expertise crosses the lifecycle: from invention to sustainment. The SEI can also rely on its deep connections to industry and academia when additional expertise is needed. …The SEI delivers value for the warfighter through a combination of line-funded research and problem-solving engagements with the DoD and other users. Using this leveraged approach enables the SEI, a relatively small institution, to accomplish highly significant results in diverse areas related to software—ranging from real-time systems design and software assurance, to Agile practices and measurement capabilities, and to secure coding and malware analytics.*
>
> —Carnegie Mellon University Software Engineering Institute Research and Development Laboratory Federally Funded Research and Development Center Comprehensive Review [DoD 2014]

Another consistent finding by these Comprehensive Reviews is that, despite the SEI's ability to address pervasive problems for the DoD, other government agencies, and their supply chains, the increasing demand for greater reliance on software, the increasing complexity of defense systems, and the expanding threats require continued innovation and even greater reliance on solutions to new problems.

## 10.6 A Vision for the Future of Software Engineering

In Section 9, the current and two former SEI chief technology officers (CTOs) collaborate in sharing their vision of the future of software engineering, particularly as it applies to the Department of Defense. All three have a long history of research in the defense, as well as the academic and industrial, contexts. This vision, strongly influenced by their personal experiences and participation in Defense Science Board, Air Force Science Advisory Board, and National Research Council studies, recognizes both the pervasiveness and criticality of software for defense. Software is both an enabler of capability and, if not carefully developed, introduces vulnerabilities that can be exploited.

The CTOs' vision emphasizes architecture-led iterative development; evidence-based software assurance and certification; critical component capabilities for cyber-physical, networked, and autonomous systems; and cybersecurity, not as separate topics so much as an interrelated collection of future capabilities. The CTO vision has consistently guided the SEI's research agenda and continues to do so today. Most of the exciting current SEI research reflects elements of this vision and promises to provide the basis for the DoD to continue relying on software for increased capability.

### 10.6.1 References

[DoD 1994] Department of Defense. *Blue-Ribbon Panel Comprehensive Review of the Software Engineering Institute.* DoD, 1994. Not publicly available.

[DoD 2004] Department of Defense. *Comprehensive Technical Review of Software Engineering Institute Programs.* DoD, 2004. Not publicly available.

[DoD 2014] Department of Defense. *Carnegie Mellon University Software Engineering Institute Research and Development Laboratory Federally Funded Research and Development Center Comprehensive Review*. DoD, 2014. Not publicly available.

# Appendix Authors Contributing to this Report

This report was written by those Members of the Technical SEI staff who were involved in the work. It was compiled by Larry Druffel who did an initial edit and is overall responsible for selection of content and presentation. Linda Pesante then edited the entire report for consistency.

The following Members of Technical Staff, current and former, contributed directly to this presentation.

| Section | Author |
|---|---|
| **Preface** | Larry Druffel |
| **Foreword** | Blaise Durante, Angel Jordan, Bob Kent |
| **Introduction** | Larry Druffel |
| **Embedded Real Time** | Dionisio de Niz, Claire Dixon, Pat Donohoe, Larry Druffel, Peter Feiler, John Foreman, John Goodenough, Mark Klein, Lui Sha, Chuck Weinstock |
| **Education and Training** | John Antonucci, Mark Ardis, Rich Caralli, Sally Cunningham, Larry Druffel, Harvey Hallman, Tom Hilburn, Chris May, Nancy Mead, Crisanne Nolan, Linda Pesante, Larry Rogers, Linda Shooer, Carol Sledge |
| **Management** | Julia Allen, Anita Carleton, Clyde Chittister, Mary Beth Chrissis, Audrey Dorofee, Larry Druffel, Jack Ferguson, Wolf Goethart, Erin Harper, Will Hayes, Mike Konrad, Steve Masters, Sally Miller, Austin Montgomery, John Morley, Jim Over, Mark Paulk, Linda Pesante, Sandy Shrum, Bob Stoddard, Bill Thomas, David White, Dave Zubrow |
| **Security** | Jeff Carpenter, Roman Danyliw, Rich Pethia, Linda Pesante |
| **Engineering Methods** | Lisa Brownsword, Peter Feiler, John Foreman, John Goodenough, Ed Morris, Tricia Oberndorf, Linda Pesante, Bill Pollak, Dennis Smith |
| **Architecture** | Len Bass, Gary Chastek, Sholom Cohen, Larry Druffel, Mark Klein, Linda Northrop, Kurt Wallnau |

| | |
|---|---|
| **Forensics** | Rich Nolan, Linda Pesante, Kris Rush, |
| **Future of Software Engineering** | Kevin Fall, Bill Scherlis, Doug Schmidt |
| **Conclusion** | Larry Druffel, John Foreman |

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704-0188*



| 1. AGENCY USE ONLY | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| (Leave Blank) | January 2017 | Final |

| 4. TITLE AND SUBTITLE | 5. FUNDING NUMBERS |
|---|---|
| A Technical History of the SEI | FA8721-05-C-0003 |

6. AUTHOR(S)
Larry Druffel

| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) | 8. PERFORMING ORGANIZATION REPORT NUMBER |
|---|---|
| Software Engineering Institute<br>Carnegie Mellon University<br>Pittsburgh, PA 15213 | CMU/SEI-2016-SR-027 |

| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) | 10. SPONSORING/MONITORING AGENCY REPORT NUMBER |
|---|---|
| AFLCMC/PZE/Hanscom<br>Enterprise Acquisition Division<br>20 Schilling Circle<br>Building 1305<br>Hanscom AFB, MA 01731-2116 | |

11. SUPPLEMENTARY NOTES

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

13. ABSTRACT (MAXIMUM 200 WORDS)

This report chronicles the technical accomplishments of the Software Engineering Institute and its impact on the Department of Defense software community, as well as on the broader software engineering community. The technical accomplishments of the SEI are interwoven with the technical developments in the broader software engineering community. The described technical work is organized into areas of importance to the mission of the SEI: Real-Time Embedded Systems, Education and Training, Management, Security, Software Engineering Methods, Software Architecture, and Computer Forensics.

| 14. SUBJECT TERMS | 15. NUMBER OF PAGES |
|---|---|
| Department of Defense, DoD, Software Engineering, Real-Time Embedded Systems, Education and Training, Management, Security, Software Engineering Methods, Software Architecture, Computer Forensics, Future of Software Engineering, Leading Software Community | 329 |

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 |

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. Z39-18
298-102