



# Are We Headed Toward a Disciplined World?

Forrest Brown  
Managing Editor



Software developers often view software as a stand-alone entity, rather than as a piece of a larger system that includes hardware and users. This can lead to many problems in system development. However, this problem can affect other people in the systems development loop, as demonstrated by the following tongue-in-cheek E-mail exchange:

**Project X team lead:** To the senior management team: Since last week's update, System X's hardware has remained on schedule, but the software problems are looking even more serious. We're averaging nearly an error per function point on the System X software, and we now project that the software is 18 weeks behind schedule, plus it's already \$220,000 over budget with an additional \$27,000-per-week burn rate. We're still waiting for your input on these and the other software problems we've been telling you about.

**Senior management:** We're glad the hardware is on schedule. Please stop bothering us with updates on software glitches. Just do whatever it takes to ensure System X meets the delivery schedule in eight weeks.

**Project X team lead:** Okay, we won't bring up software problems with you anymore. By the way, we have refigured the system schedule: The *system* will be delivered at least 18 weeks late, and

the *system* is already \$220,000 over budget and counting. Please comment.

**Senior management:** We need an emergency meeting right away. Why didn't you tell us earlier that the system was in trouble?

Efforts to implement better practices and more mature processes have been going on for years now in the Department of Defense (DoD). The Capability Maturity Model (CMM) for Software was developed to "provide software organizations with guidance on how to gain control of their processes for developing and maintaining software and how to evolve toward a culture of software engineering and management excellence" (CMM for Software, Version 1.1, CMU/SEI-93-TR-24, p. 5). In the DoD, it has been the model of choice for software and has improved cost, quality, and schedule performance in organizations where it has been skillfully applied.

Unfortunately, most software organizations still have a long way to go before they will be mature and disciplined enough to avoid the self-inflicted problems that are wrongly assumed to be a natural part of software development. The same applies to the problems caused by the widespread lack of discipline in systems development.

Many improvement efforts are now ongoing in the various engineering disciplines related to systems development. Early indicators from organizations that use the resulting models reveal that the application of the practices defined in these models leads to more discipline, which results in improved schedule, cost, and quality.

Each of the past and present models and standards attempts to move system development toward more disciplined, mature behavior. To help you keep track of all these developments, Randall Wright's article (p. 7) eliminates some of the confusion regarding different systems engineering models that have evolved since process improvement efforts began.

*CROSSTALK* has long discussed processes and process models for software. Most of the articles in this issue discuss software in the context of the larger system. We hope it will help you better understand DoD's and industry's game plan for future success in systems development. Slowly but surely, we should hope to see the day when most organizations produce systems and systems software with greater discipline and maturity. ♦



## Software Metrics Hazards

Elizabeth Starrett's article, "Measurement 101," *CROSSTALK*, August 1998, was interesting and well written, but it left out a critical point. Metrics based on "source lines of code" move backward when comparing software applications written in different programming languages. The version in the low-level language will look better than the version in the high-level language.

In an article aimed at metrics novices, it is very important to point out

some of the known hazards of software metrics. The fact that lines of code can't be used to measure economic productivity is definitely a known hazard that should be stressed.

In a comparative study of 10 versions of the same period using 10 different programming languages (Ada 83, Ada95, C, C++, Objective C, PL/I, Assembler, CHILL, Pascal, and Smalltalk), the lines of code metric failed to show either the highest pro-

ductivity or best quality. Overall, the lowest cost and fewest defects were found in Smalltalk and Ada95, but the lines of code metric favored assembler. Function points correctly identified Smalltalk and Ada95 as being superior, but lines of code failed to do this.

Capers Jones  
*Software Productivity Research*



# Software Engineering and Systems Engineering in the Department of Defense

Mark Schaeffer

*Office of the Undersecretary of Defense for Acquisition and Technology*

*As the Department of Defense (DoD) moves toward acquisition without rigid DoD standards, acquisition is becoming less an issue of hardware vs. software or software engineering vs. systems engineering—it is becoming an integrated whole. New DoD directives, an integrated Capability Maturity Model, highly trained acquisition personnel, and a partnership with industry to manage risk, will help us meet the challenges of acquisition in the next century.*

Twenty years ago, complex machines were controlled by hydraulic actuators or mechanical linkages using position switches and analog control circuitry. Today, much of the mechanical or hydraulic equipment and almost all analog circuitry has been replaced by servo motors, digital position encoders, and microprocessors. Complex machines still have a large mechanical component, but their control is often digital rather than analog. Presently, every DoD development project involves digital technology at some level in the work breakdown structure—either digital control systems, automated test equipment, or training devices. Software engineering has become a critical and key component of the engineering development process for both military and commercial product development.

Over the past 20 years, our systems engineering policy and practice has evolved alongside the revolution in digital technology. However, as digital technology evolved in its importance in control systems, the software needed to make them work properly became more problematic. Our track record using DoD Directive (DoDD) 5000.1 for weapons systems acquisition and DoDD 8120.1 for information systems acquisition was not good and was not getting better. In 1991, the DoD published its first acquisition reform study, and in June 1993, Deputy Undersecretary of Defense for Acquisition Reform Colleen Preston opened the DoD's acquisition reform office. Prior to the official start of acquisition reform, studies of software engineering noted frequent and serious schedule, cost, and performance slippages. Figure 1 illustrates the issues.

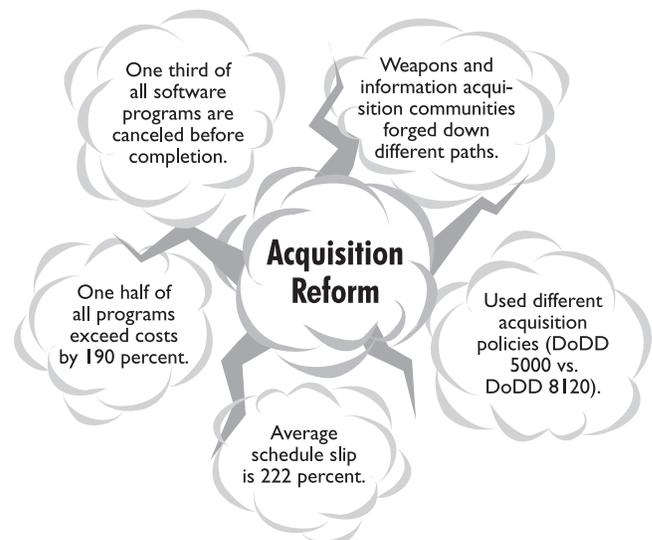
As acquisition reform gained momentum, the DoD embraced a key conceptual change in policy. Instead of developing and procuring hardware and software using different processes, the DoD sought to acquire weapons systems that met stated performance objectives. We moved away from telling contractors how to build a product and toward defining the end performance of a product we would buy. We sometimes refer to this as the Performance-Based Business Environment. We no longer require that electronic components meet and be tested in accordance with rigid military standards. Instead, we asked developers to show us that their products could operate over a military

operational mission profile, could be effectively and efficiently maintained, and would have the required operational life span. Commercial design and development practices are encouraged, and blind adherence to military standards is not allowed. This shift of philosophy was influenced by another reality of the 1990s: The number of new-start major programs and new-start non-major programs was decreasing dramatically when compared to the 1970s and 1980s.

Acquisition reform provided the impetus and the structure for the DoD to take the initial step toward incorporating software engineering into the larger context of systems engineering. But was not software engineering unique or at least fundamentally different from the rest of the development process?

Our analysis confirmed our belief that the objectives of software engineering, the problems encountered managing it, and the techniques used to resolve those problems are essentially the same as those found in systems engineering. For example, the management of risk in software-intensive programs is fundamentally the same as for any other type of program. This similarity led us to combine the treatment that software and hardware risk management receives in

Figure 1. *The pre-acquisition reform environment.*



Managing the Job	Developing Product	Assuring Product
<ul style="list-style-type: none"> <li>• Integrated Product and Process Development (IPPD)</li> <li>• Planning and Estimating</li> <li>• Work Breakdown Structure</li> <li>• Contracting</li> <li>• Personnel and Resources</li> <li>• Integrated Program Management</li> <li>• Engineering Management</li> <li>• Requirements Management</li> <li>• Interface Management</li> <li>• Configuration Management</li> <li>• Risk Management</li> <li>• Policy</li> <li>• Licenses</li> <li>• Logistics</li> <li>• Training</li> </ul>	<ul style="list-style-type: none"> <li>• IPPD</li> <li>• Product Lines</li> <li>• Modeling and Simulation</li> <li>• Design</li> <li>• Integration</li> <li>• Trade-off Studies</li> <li>• Open Systems</li> <li>• Continuous Process Improvement</li> <li>• Reliability</li> <li>• Interoperability</li> <li>• Producibility</li> <li>• Maintainability</li> <li>• Non-Developmental Item</li> <li>• Government-off-the-Shelf Software and Commercial-off-the-Shelf Software</li> </ul>	<ul style="list-style-type: none"> <li>• IPPD</li> <li>• Quality</li> <li>• Verification and Validation</li> <li>• Inspections</li> <li>• Measurement</li> <li>• Tracking</li> <li>• Sustainment</li> <li>• Supportability</li> <li>• Traceability</li> <li>• Test and Evaluation</li> <li>• Safety</li> </ul> <p><b>Special Considerations</b></p> <ul style="list-style-type: none"> <li>• Architecture</li> <li>• Programming Languages</li> <li>• Modification and Upgrade</li> <li>• Post-Deployment Support</li> </ul>

Figure 2. *Elements of systems acquisition.*

the Defense Acquisition Deskbook. Figure 2 shows the activities that are common to systems engineering and software engineering.

We did find a few software engineering activities that deserve special consideration, but dealing with them in the systems engineering context is not difficult. We found that

- Architecture selection requires a global approach early in the design process because these architecture selections affect many other design choices downstream and are costly to reverse.
- With relaxation of the Ada mandate, there is a greater choice of development languages, which increases program risk. Mitigating this risk is accomplished using rigorous business case analyses for language selection.
- As hardware systems age, they generate a larger pool of trained maintainers; however, rapid technological change reverses this process for software. Searching for programmers to meet Year 2000 demands is an example of a work force that although technologically advanced is not able to meet post-deployment support needs.
- Post-deployment support is common to both hardware and software but different in its application.

There is no software equivalent of a form-fit-function replacement using a different product. Similar to post-deployment support, modification issues are not unique to software. Although software modification holds the promise of greater and easier system improvement, it has not, in practice, generated the promised efficiencies and

benefits. Software is not as malleable as originally believed, nor are “tweaks” necessarily as low risk and inexpensive as hoped.

Dealing with each of these areas is not beyond our ability; the key lies in early planning and having a risk reduction effort. Once we saw the significant degree of commonality between the elements of systems engineering and software engineering, we set out to deal with both of them in DoD acquisition policy.

In March 1996, the DoD created a single acquisition process that “states the policies and principles for all DoD acquisition programs ...” when it issued DoDD 5000.1, Defense Acquisition, and DoD Regulation 5000.2-R, Mandatory Procedures for Major Defense Acquisition Programs and Major Automated Information System Acquisition Programs. It also canceled DoDD 8120.1, Life-Cycle Management of Automated Information Systems, and provided program managers the means to manage their programs using a total systems approach, to optimize total system performance, and to minimize total ownership costs. Our belief that systems engineering and

Figure 3. *DoD acquisition management key policies.*

**DoDD 5000.1 Defense Acquisition**

“Software is a key element in DoD systems. It is critical that software developers have a successful past performance record, experience in the software domain or product line, a mature software development process, and evidence of use and adequate training in software methodologies, tools, and environments” (DoDD 5000.1, para. 2.k).

**DoDD 5000.2 Mandatory Procedures for Major Defense Acquisition Programs and Major Automated Information Systems Acquisition Programs**

“Software shall be managed and engineered using best processes and practices that are known to reduce cost, schedule, and performance risks. It is DoD policy to design and

develop software systems based on systems engineering principles ...” (DoD 5000.2-R, para. 4.3.5).

To include

- Developing software system architectures that support open system concepts.
- Exploiting COTS products.
- Identifying and exploiting software reuse.
- Selecting a programming language in the context of systems and software engineering factors (ASD (C31) memo, April 29, 1997).
- Use of DoD standard data (DoDD 8320.1).
- Selecting contractors with
  - Domain experience in comparable systems.
  - Successful past performance record.
  - Demonstrable mature software process.
- Use of software metrics.
- Assessing information warfare risks IAW DoDD TS-3600.1.

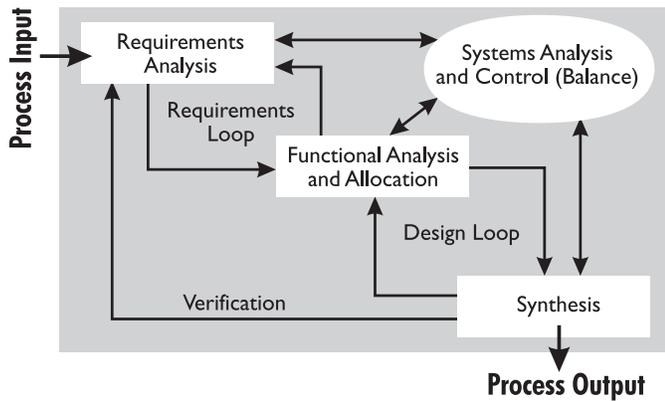


Figure 4. *The systems engineering process.*

software engineering are integral was recently validated when we updated DoD Regulation 5000.2-R to comply with the Clinger-Cohen Act. Most of the revisions were not technical in nature but consisted of adding to existing sections of the regulation references to the Act. Figure 3 summarizes how we implemented provisions of the Clinger-Cohen Act and integrated software engineering policy into the framework of systems engineering and major program acquisition.

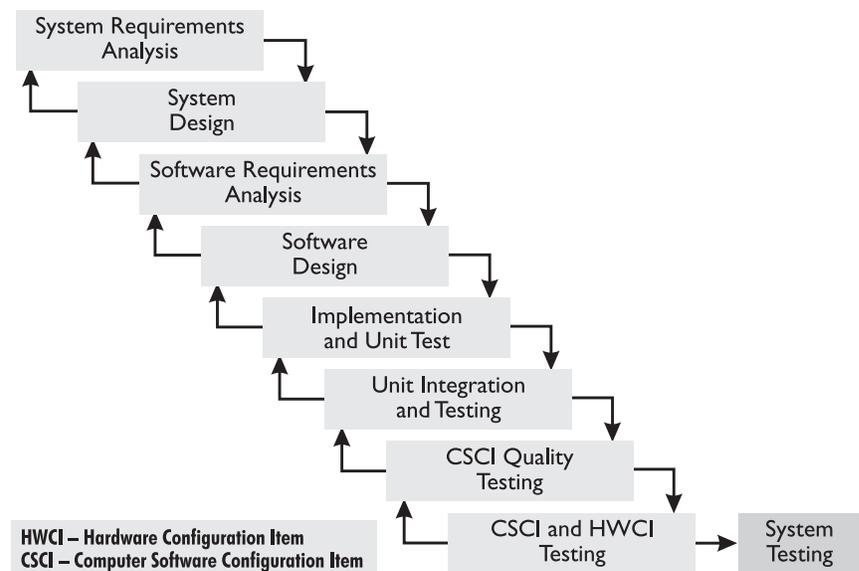
So, how does the software development process integrate into the systems engineering process? There are certain aspects of the systems engineering process that are highly correlated, while others are more difficult to match up. The system engineering process, as addressed in the Defense Acquisition Deskbook, is comprised of four elements:

- Translating stated operational requirements into an integrated product design using a systematic, concurrent approach.
- Transitioning multidisciplinary technical inputs (including concurrent engineering of manufacturing, logistics, and testing) into a coordinated effort to meet program cost, schedule, and performance objectives.
- Ensuring functional and physical interface compatibility so system definition and design meet all hardware, software, facilities, people, and data requirements.
- Establishing a risk management program to reduce risk early through system element tests and demonstrations.

Software engineering management uses both sequential and nonsequential processes that correspond to activities in the systems engineering process. The waterfall model, shown in Figure 5, is a sequential process that has the following characteristics:

- Do the steps in a specified order.
- Define all the requirements upfront.
- Use comprehensive reviews as gates.

Figure 5. *The waterfall model.*



- Complete program design before coding.
- Emphasizes functional and allocated baselines.
- “Do the job twice if possible and involve the customer.”

Figure 6 illustrates how steps in the waterfall model correspond to iterations of the systems engineering model.

There are several other models for software lifecycle management, and they all can be mapped to the iterative systems engineering process—even those models designed to handle ill-defined user requirements.

Our conclusion, reached several years ago, and confirmed repeatedly since then, is that software engineering is an integral part of systems engineering. The systems engineering management techniques we have honed and the process improvement efforts we have undertaken consolidate well with similar processes and improvement efforts in software engineering. We are about to embark on the next step in recognizing software engineering as an integral part of the systems engineering process.

### Where Will We Focus Our Future Effort?

First, we are engaged in a dialog with the command, control, communications, and intelligence community about consolidating regulations that affect information technology acquisition. There are a number of duplicative and overlapping regulations that have their genesis in the days when there were different “stovepipes” for weapons systems and automated information system acquisition. As the information revolution progresses and gains momentum, virtually all acquisition will contain information technology. Our intent is to derive the most synergism we can from the DoD’s implementation of the Clinger-Cohen Act while making good on our commitment to provide clear, unambiguous, and realistic guidance to program managers.

Second, to take advantage of the information revolution, we must have an acquisition work force that has the

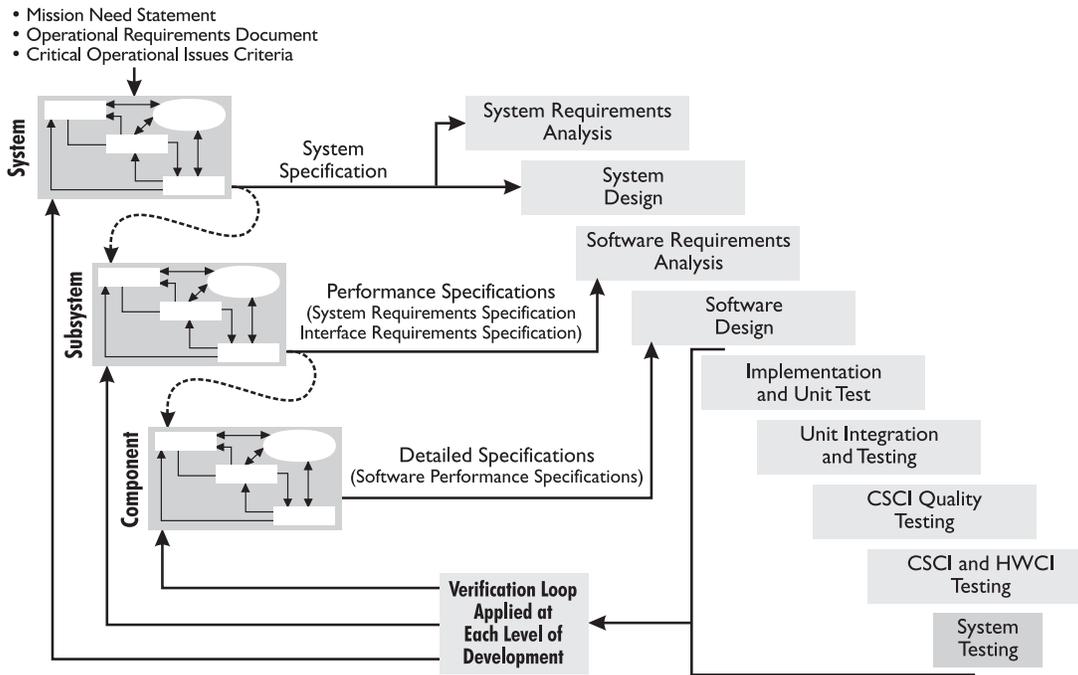


Figure 6. *The relationship between software engineering and systems engineering.*

education and training to manage complex integrated hardware and software engineering activities. We have initiated a comprehensive upgrade of the training materials used by the Defense Acquisition University for acquisition work-force training. This upgrade will incorporate software engineering management into the resident, nonresident, and distance learning curriculum. Our goal is to have an acquisition work force that is capable of maximizing the benefit of the process improvements we are putting in place.

Third, we are in a partnership with industry to develop new practices, procedures, and techniques for the management and reduction of risk of complex weapons systems acquisitions. The National Defense Industrial Association is supporting extensive work in this area, and it holds the promise of big dividends helping the DoD complete the transition from the “how to” of military specifications and standards to the performance-oriented environment of acquisition reform.

Last but not least, we are engaged in a comprehensive integration of Capability Maturity Models (CMM)

originally developed by the Software Engineering Institute, and now being championed by industry at large. The initial common framework effort will be based on the software CMM, the systems engineering capability model, and the Integrated Process Development (IPD) CMM. Other functional disciplines may be added later. The work accomplished to date on the software CMM, Version 2.0, and the IPD CMM have been included in the initial CMM Integration baseline. The goal is to improve efficiency, return on investment, and effectiveness by using models that integrate disciplines such as systems engineering and software engineering—disciplines that are inseparable in a systems development endeavor.

As we cross the millennium, we are committed to developing and implementing a complete framework for the management of acquisition within the DoD. The days of separate hardware and software acquisition is gone. The challenge of creating a new, integrated systems engineering and software engineering framework is a daunting one, but we are up to the task. ♦

#### About the Author



**Mark Schaeffer** is the deputy director for systems engineering in OUSD(A&T). He is responsible for policy and implementation of systems engineering, technical risk management, manufacturing, quality, reliability and maintainability, acquisition logistics, modeling and simulation, and software engineering.

Schaeffer has over 20 years experience in weapons systems acquisition and program management in the Office of the Secretary of Defense, Naval Sea Systems Command, and as congressional staff. He has served in several challenging management positions within the Naval Sea Systems Command, to include deputy program manager and technical director of the MK-48/ADCAP Torpedo Project; program manager of the CV/Amphib Firefighting Improvement Program; and special assistant to the Shipyard Planning officer and chief design engineer at Mare Island Naval Shipyard.

Schaeffer has a bachelor's degree in mechanical engineering from California State University at Sacramento and has completed graduate studies at Massachusetts Institute of Technology, Duke University, and Georgetown University.

E-mail: mschaeff@acq.osd.mil

# Process Standards and Capability Models for Engineering Software-Intensive Systems

Randall R. Wright  
Software Technology Support Center

*Current standards and models have improved the quality, cost, and repeatability of systems engineering products and processes. However, soon-to-be-published documents are the next step in developing, maintaining, and reengineering large, complex, software-intensive systems. These efforts consolidate existing documents and minimize the impact of transitioning your process improvement activities. This article explains the changes and how they affect you.*

Fortunately for systems engineering (SE), the recent changes in process standards and capability models are for the better. In addition to documenting and expanding our body of knowledge, the SE community is combining the efforts of several agencies into consolidated documents (Figure 1). These emerging “best practices” will show forward-looking organizations how to stay competitive in our ever-changing field. Of special interest is the expanding role software engineering plays in systems engineering.

## SE Process Standards

Current and emerging standards on how to engineer a system, although similar, have varied scopes (Figure 2). Their intended audience, e.g., manager, practitioner, determines the level of detail and breadth of coverage. You may choose the standard that best meets your needs or, with the emerging standards, choose only the processes that apply to you.

## Current Standards

MIL-STD-499B, *Systems Engineering Management*; EIA Interim Standard 632, *Processes for Engineering a System*; and Institute of Electrical and Electronics Engineers (IEEE) 1220-1994, *Application and Management of the Systems Engi-*

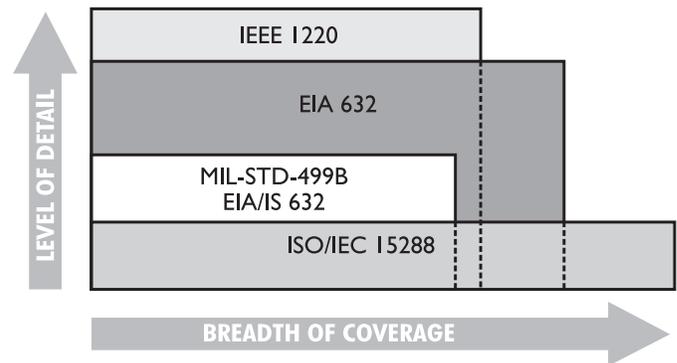


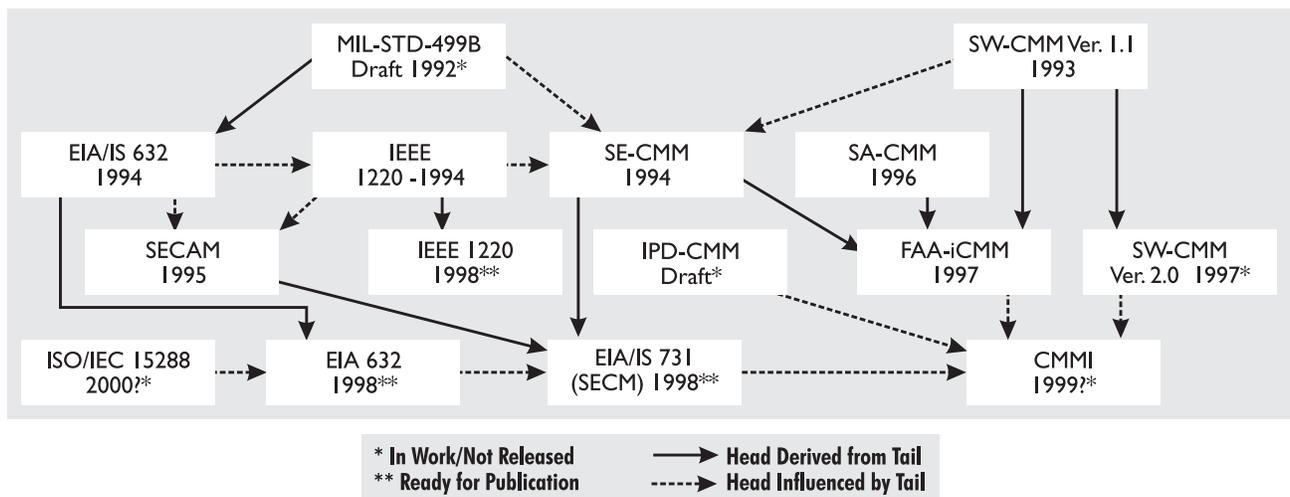
Figure 2. Scope of SE process standards.

neering Process all cover one SE process, which consists of the following:

- Requirements analysis.
- Functional analysis.
- Synthesis.
- Systems analysis.
- Control.

You may apply any of the versions of this process to the development or modification of a system. These standards require specific tasks for each activity of the process and

Figure 1. Relationships of SE standards and models.



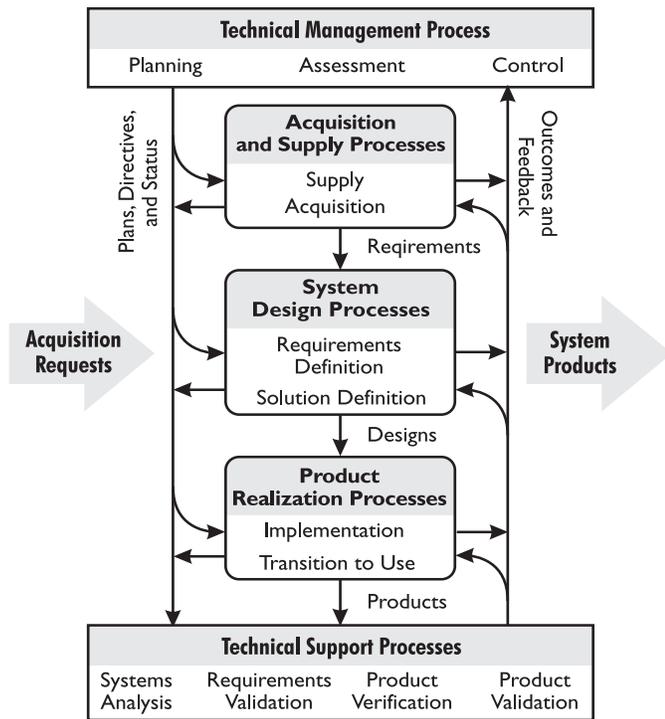


Figure 3. EIA 632 processes.

the use of a detailed management plan and event-based and time-based schedules.

**Emerging Standards**

The upcoming IEEE 1220 will change little from the “trial use” 1220-1994. Electronics Industry Association (EIA) 632, *Processes for Engineering a System*, expands on previous work and will be the basis for implementation of International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) 15288, *System Life Cycle Processes*, in the United States. EIA 632 has 13 technical and project processes (Figure 3) that cover

- Acquisition and supply.
- System design.
- Product realization.
- Technical management.
- Technical support.

The working draft of ISO/IEC 15288 currently has 22 generic processes that address enterprise-wide issues and technical and project concerns (Figure 4). With EIA 632, you apply the appropriate processes (each consisting of one to five requirements along with recommended tasks and expected outcomes) for the top-down design of system products as well as the bottom-up realization of such products (Figure 5). With ISO/IEC 15288, you choose the processes you

need (each consisting of two to five activities with recommended tasks) to meet specified lifecycle requirements of a software-intensive system. This process goes down to, but does not include, the software.

**Process Standards for Software**

MIL-STD-498, *Software Development and Documentation* was canceled May 27, 1998. Its replacement is IEEE/EIA 12207, *Information Technology – Software Life Cycle Processes*, the U.S. implementation of ISO/IEC 12207. Reportedly, but unconfirmed, the commercial interim standard J-STD-016-1995, *Software Life Cycle Processes for Software Development* (derived from MIL-STD-498) is not going away and should be finalized (J-STD-016) sometime next year.

**SE Capability Models**

Current and emerging capability models for systems engineering aim to repeat the benefits of the Software Engineering Institute’s (SEI) *Capability Maturity Model for Software* (SW-CMM):

- Better competitive position.
- Returns on investment of between 4.5 and 7.7-to-1 (as have been experienced by Hughes, Tinker Air Logistics Center, and Raytheon).
- Predictable and reduced cost and schedule.
- Reduced risks and fewer trouble reports.
- Improved customer satisfaction and employee morale.
- Less overtime, absenteeism, and turnover.

In addition, integrated SE and software models should save time and money and reduce redundancy in assessments for both software and SE process improvement. Fortunately, the models map well to each other (Figure 6). Even at lower levels of detail, the models specify similar functions. Improvement efforts based on older models will not be wasted, and the transition to a newer model should not be traumatic. For example, if winning a contract de-

Figure 4. ISO/IEC 15288 processes (working draft).

<p><b>Agreement Processes</b></p> <ul style="list-style-type: none"> <li>Acquisition</li> <li>Supply</li> <li>Negotiation</li> </ul>	<p><b>Project Management Processes</b></p> <ul style="list-style-type: none"> <li>Planning</li> <li>Assessment</li> <li>Control</li> </ul>
<p><b>Enterprise Processes</b></p> <ul style="list-style-type: none"> <li>Investment Management</li> <li>Multi-Project Management</li> <li>Enabling Infrastructure</li> <li>Human Resources</li> <li>Process Management</li> <li>Quality Management</li> <li>Risk Management</li> </ul>	<p><b>Technical Processes</b></p> <ul style="list-style-type: none"> <li>Acquirer Requirements Definition</li> <li>Other Stakeholder Requirements Definition</li> <li>System Requirements Definition</li> <li>System Architecture Design</li> <li>System Architecture Design Implementation</li> <li>System Product Validation</li> <li>System Product Verification</li> <li>System Product Transition</li> <li>Systems Analysis</li> </ul>

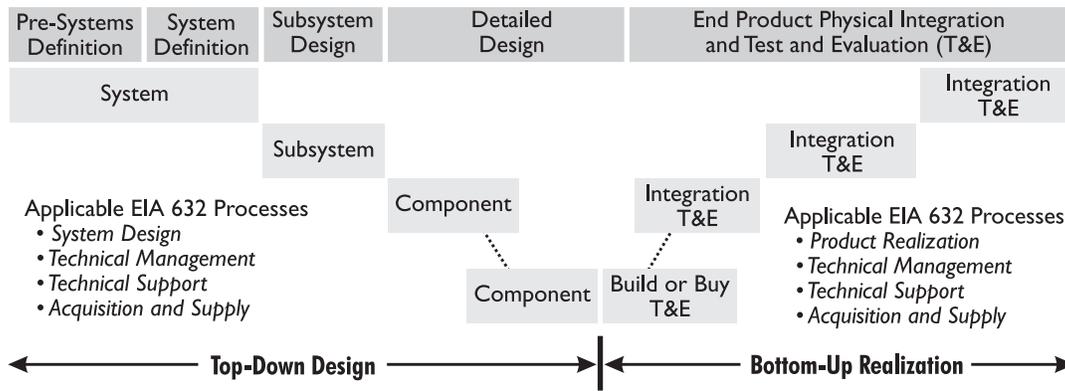


Figure 5. Engineering lifecycle.

depends on an evaluation using a newer model, most of what you have already done should “find a home” under a new name.

**Current Models**

The *Systems Engineering Capability Maturity Model* (SE-CMM), was published by the Enterprise Process Improvement Collaboration in 1994. About the same time, the International Council on Systems Engineering (INCOSE) developed the *Systems Engineering Capability Assessment Model* (SECAM). Although SECAM has less visibility than SE-CMM, both are being used in the SE community. Anecdotal evidence from those who have used the SE-CMM suggests a return on investment similar to software CMM use. Lockheed Martin has reported “a positive difference” from “more mature systems engineering processes.”

Last year, the Federal Aviation Administration (FAA) published its *Integrated Capability Maturity Model* (FAA-iCMM<sup>SM</sup>), which combines the software, the systems engineering, and the software acquisition CMMs into one integrated model. FAA uses this

in-house and freely distributes it. Plans are under way for three divisions at Warner-Robins Air Logistics Center (software, SE, and acquisition) to use FAA-iCMM and Integrated Process and Product Development (IPPD) as guides for “enterprise-wide process improvement.”

**Emerging Models**

EIA Interim Standard 731, *Systems Engineering Capability Model* (SECM), provides complete coverage of EIA 632 and is consistent with IEEE 1220. SECM (a merging of SE-CMM and SECAM) has 19 focus areas that address technical, management, and environment issues (Figure 7). The future of this interim standard depends on the National Defense Industrial Association’s *Capability Maturity Model Integration* (CMMI) effort. If CMMI successfully incorporates SECM concepts, EIA Interim Standard 731 would be duplicative and would probably be rescinded. Otherwise, the SECM will progress to a full (vs. interim) standard.

CMMI will provide a common framework for multiple capability models. In its first version, CMMI

will integrate SECM, the software CMM, and Integrated Product Development CMM (IPD-CMM) concepts and build on the FAA-iCMM effort. The result will be a core of common processes and additional domain-specific processes for software and for SE (Figure 8). Reportedly, there is much commonality between the three models and few domain-specific processes. CMMI’s first version will give

Figure 7. EIA interim standard 731 focus areas.

**Technical**

- 1.1 Define Stakeholder and System Level Requirements
- 1.2 Define Technical Problem
- 1.3 Define Solution
- 1.4 Assess and Select
- 1.5 Integrate System
- 1.6 Verify System
- 1.7 Validate System

**Management**

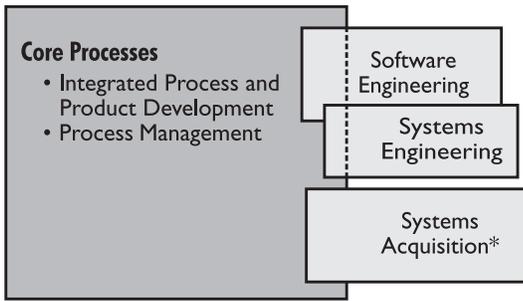
- 2.1 Plan and Organize
- 2.2 Monitor and Control
- 2.3 Integrate Disciplines
- 2.4 Coordinate with Suppliers
- 2.5 Manage Risk
- 2.6 Manage Data
- 2.7 Manage Configurations
- 2.8 Ensure Quality

**Environment**

- 3.1 Define and Improve the Systems Engineering Process
- 3.2 Manage Competency
- 3.3 Manage Technology
- 3.4 Manage SE Support Environment

Figure 6. Mapping of SE capability models.

SE-CMM	SECAM	SECM	FAA-iCMM
Engineering	Systems Engineering	Technical	Lifecycle or Engineering Supporting (Not Lifecycle Dependant)
Project	Management	Management	Management or Project
Organization	Organization	Environment	Organizational



\*Initial release will not include an acquisition model.

Figure 8. CMM Integration.

three models from which to choose.

You may

- Conduct a software assessment using core processes and software processes.
- Conduct an SE assessment using core processes and SE processes.
- Conduct an integrated assessment using core processes and combined software and SE processes.

The next version of the CMMI is likely to incorporate the software acquisition CMM (SA-CMM). Subsequent versions may address additional models (Secure Systems Engineering CMM, People CMM, Team CMM, etc.)

### Capability Models for Software

The CMM for Software (versions 1.0 and 1.1) has seen wide use and acceptance since 1993. SEI has halted the nearly complete update (version 2.0, draft C) in anticipation of CMMI (described above).

### Development of Some Models Placed On Hold

Some organizations are so interested in an integrated capability model they are developing their own in-house versions, as the FAA did. However, since CMMI seems imminent, Litton PRC and Rockwell/Collins (and probably others) have halted such efforts. Likewise, SEI will not release version 2.0 of the SW-CMM. It also is uncertain whether the FAA will update its iCMM as planned. Finally, as reported above, EIA Interim Standard 731 (SECM) is not currently being considered for publication as a full standard.

### Impetus for Change

You have seen how the SE climate is changing; current standards and models are giving way to better ones. You know the benefits of improving your business; staying competitive is imperative. Structured process improvements are the key to successful adoption of these new technologies.

- “If you don’t know where you are, a map won’t help.” – *Watts Humphrey*
- “If you don’t know where you are going, any road will do.” – *Chinese proverb*
- “Even if you’re on the right track, you’ll get run over if you just sit there.” – *Arthur Godfrey*

### Acknowledgments

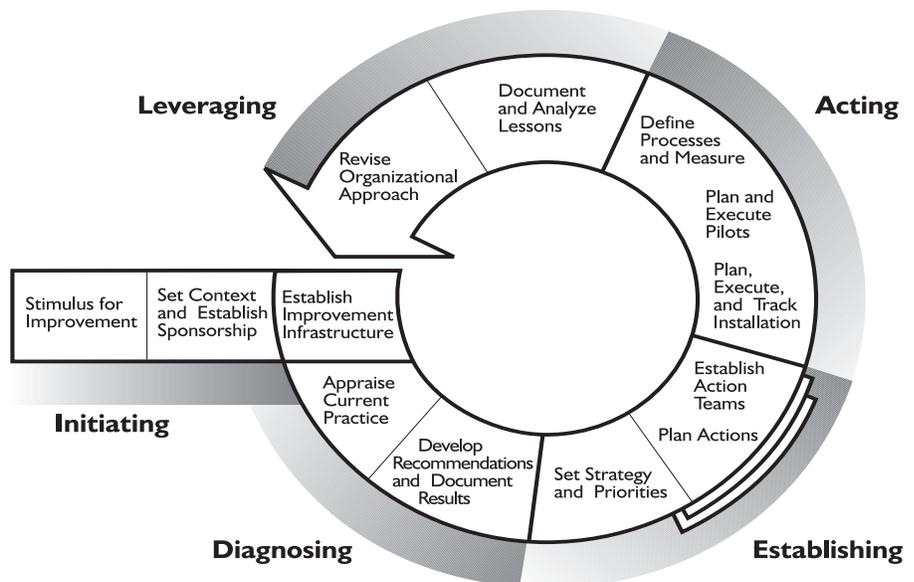
I thank the following subject matter experts, whose presentations at the Eighth International Symposium of INCOSE and discussions were the source for the majority of the information and graphics in this article: Don Barber and Bill Mindlin, chairmen of the INCOSE Capability Assessment Working Group; Lt. Col. Joe Jarzombek, director of the U.S. Air Force Embedded Computer Resources Support Improvement Program; Jerry Lake, owner and chief scientist of Systems Management International; Sarah Sheard, senior systems engineer at the Software Productivity Consortium. I also thank their respective organizations for allowing the modification of their graphics for this article. ♦

### About the Author



**Randall R. Wright** is a consultant at the Software Technology Support Center (STSC) specializing in systems engineering products and services, representing the STSC on the CMMI project, and coordinating the next version of *Guidelines for Successful Acquisition and Management of Software-Intensive Systems*. He has over 20

Figure 9. IDEAL model.



years military and civilian service with aircraft, missiles, and various anti-armor munitions and nuclear delivery systems. He has performed at various levels in maintenance, acquisition, logistics, operations, and policy. He has a bachelor's degree in electrical engineering from Arizona State University, completed Air University's Software Professional Development Program and Air Command and Staff College, holds three acquisition professional certifications, and is a captain in the Air Force Reserves.

Software Technology Support Center  
7278 Fourth Street  
Hill AFB, UT 84056  
Voice: 801-777-9732 DSN 777-9732  
Fax: 801-777-8069 DSN 777-8069  
E-mail: [wrightr@software.hill.af.mil](mailto:wrightr@software.hill.af.mil)  
Internet: <http://www.stsc.hill.af.mil/>

## Document Sources

Copies of the documents discussed in this article can be obtained from the following sources.

### Capability Maturity Models

SEI Customer Relations  
4500 Fifth Avenue  
Pittsburgh, PA 15213  
Voice: 412-268-5800  
E-mail: [customer-relations@sei.cmu.edu](mailto:customer-relations@sei.cmu.edu)  
Internet: <http://www.sei.cmu.edu/pub/documents/.../96.reports/pdf/tr020.96.pdf> (SA-CMM)  
[.../96.reports/pdf/hb004.96.pdf](http://www.sei.cmu.edu/pub/documents/.../96.reports/pdf/hb004.96.pdf) (SE-CMM)  
[.../94.reports/pdf/tr24.94.pdf](http://www.sei.cmu.edu/pub/documents/.../94.reports/pdf/tr24.94.pdf) (SW-CMM)

### FAA-iCMM

Federal Aviation Administration  
AIT-5, 800 Independence Avenue SW  
Washington, DC 20591  
Voice: 202-267-7443  
E-mail: [linda.ibrahim@faa.dot.gov](mailto:linda.ibrahim@faa.dot.gov)  
Internet: <http://www.faa.gov/ait/ait5/FAA-iCMM.htm>

### IEEE 1220, IEEE/EIA 12207

IEEE Service Center  
P.O. Box 1331  
Piscataway, NJ 08855-1331  
Voice: 800-678-4333  
E-mail: [customer.service@ieee.org](mailto:customer.service@ieee.org)  
Internet: <http://standards.ieee.org>

### DoD agencies may obtain copies at

Standardization Order Desk  
700 Robbins Avenue, Building 4/D  
Philadelphia, PA 19111-5094

### SECAM

INCOSE, 2033 Sixth Avenue #804,  
Seattle, WA 98121

## Successfully Adopting New Technologies

Adopting a new technology (whether it is a process, methodology, or tool) means *change!* This is more than a technical issue; you must overcome social, behavioral, managerial, and organizational barriers. To quote Capers Jones, "As a general rule, technology [adoptions] are not very rapid processes, and the bigger the organization, the longer it takes."

When adopting new technologies, the STSC recommends implementing them with the aid of the SEI's IDEAL Model (Figure 9).

- Initiating "sets the stage" for process improvement by stimulating change and laying the necessary groundwork.
- Diagnosing is the gap analysis to see where you are, decide where you want to be, and determine what you will do next.
- Establishing puts the people and plans in place.
- Acting is the execution of your plans with appropriate measures and project tracking.
- Leveraging lets you learn from what you did and do better the next time.

Voice: 800-366-1164, 206-441-1164  
E-mail: [incose@halcyon.com](mailto:incose@halcyon.com)  
Internet: <http://www.incose.org/priclist.html>

EIA 632, EIA 731 ISO/IEC 12207,  
ISO/IEC15288, J-STD-016, MIL-  
STD-498, MIL-STD-499B  
(commercial source)

Global Engineering Documents  
15 Inverness Way East  
Englewood, CO 80112-5776  
Voice: 800-854-7179  
E-mail: [global@his.com](mailto:global@his.com)  
Internet: <http://global.his.com>

### Recommended Reading

1. Hollenbach, Craig R. and Steve Mosier, "Developing and Applying a Joint SW-CMM and SECM Appraisal Methodology," *Eighth Annual International Symposium of INCOSE*, Vancouver, British Columbia, Canada, July 26-30, 1998.
2. Jarzombek, Lt. Col. Joe, "CMMI Supports Enterprise-Wide Process Improvement," *CROSSTALK*, July 1998, p. 2.
3. Lake, Jerome G., "Processes for Engineering Systems," *Eighth Annual International Symposium of INCOSE*, Vancouver, British Columbia, July 26-30, 1998.
4. Martin, James N., "Evolution of EIA 632 from an Interim Standard to a Full Standard," *Eighth Annual International Symposium of INCOSE*, Vancouver, British Columbia, July 26-30, 1998.
5. Martin, James N., "Overview of the EIA 632 Standard: Processes for Engineering a System," *Eighth Annual International Symposium of INCOSE*, Vancouver, British Columbia, July 26-30, 1998.
6. "Military Standard Software Development and Documentation," *CROSSTALK*, August 1998, p. 4.
7. Schaeffer, Mark D., "Capability Maturity Model Process Improvement," *CROSSTALK*, May 1998, p. 3.
8. Schaeffer, Mark D., Philip Babel, Jack Ferguson, et al., "Overview of the Integrated Capability Maturity Model (CMMI) Development Project," panel presentation, *Tenth Annual Software Technology Conference*, Salt Lake City, Utah, April 22, 1998.
9. Sheard, Sarah A., "Quagmire," [www.software.org/quagmire](http://www.software.org/quagmire).
10. Sheard, Sarah A. and Jerome G. Lake, "Systems Engineering Standards and Models Compared," *Eighth Annual International Symposium of INCOSE*, Vancouver, British Columbia, July 26-30, 1998.
11. Sheard, Sarah A., Susan Rose, Linda Inrahim, and Wafa Makhlof, "Two Approaches to CMM Integration," *Eighth Annual International Symposium of INCOSE*, Vancouver, British Columbia, July 26-30, 1998.
12. Widmann, E. R., "Key Features of the "Merged" EIA/IS 731-1 Systems Engineering Capability Model," *Eighth Annual International Symposium of INCOSE*, Vancouver, British Columbia, July 26-30, 1998.

# Designing a Digital Library to Support Global Army Training

Lt. Col. Anthony Ruocco and Maj. Todd L. Smith  
*U.S. Army*

In 1995, the Army's Training and Doctrine Command (TRADOC) and the Army National Guard began to invest in Key Enabling Initiatives to provide "training to the soldier any time and anywhere in the world." They had the additional goal to use appropriate current media formats, migrating to superior media as the technology becomes available. Computer-based training promises to provide soldiers high-quality training while reducing costs and travel time.

In their *Army Training XXI Campaign Plan*, the deputy chief of staff for training at TRADOC has created three Axes of Attack to integrate advanced technology into the Army Training Plan. These axes include Warfighter XXI for unit training, Warrior XXI for institutional and self-development, and Warnet XXI, which provides support for the other axes. The electronic foundation is the Army Training Digital Library, which can be accessed at <http://www.atsc-army.org/atdls.html>. This digital library allows users to access a wealth of training materials and resources stored on Digital Training Access Centers at TRADOC installations all over the country. For example, the access center at Fort Knox, Ky. would provide primary support for the Armor School while Fort Rucker, Ala. would support Aviation. There are 21 schools on 15 Army posts, so 15 access centers would exist to distribute training to a globally deployed Army. This provides a distributed database that

will improve user accessibility and reduce network congestion.

TRADOC has fielded two operational access centers. One, located at Fort Eustis, Va., uses an Asynchronous Transfer Mode network. The other is located at the United States Military Academy at West Point and uses switched fast Ethernet.

## Architecture

The Digital Training Access Center consists of an integrated Web server, database server, and video server. The Web and database server can run on NT workstations with a minimum of two processors. Clustering technologies appear promising and could offer additional scalability and fault tolerance. In order to provide concurrent, streamed video to a large student population (more than three classrooms), the access center requires a Video Server. Video Servers can be NT- or Unix-based, tend to be highly scalable, and provide robust connectivity. A general-purpose file server can provide file-based video for a small student population.

These solutions are intended for a high-bandwidth Intranet. For World Wide Web or Internet distribution, the access center would require a low-bandwidth alternative. The systems currently fielded at Fort Eustis and West Point can provide low-bandwidth (28.8 Kbps) access to remote users and high-bandwidth (up to 3 Mbps) access to local students. Remote users can obtain higher-quality video through compact disc or digital video disc distribution, but version control

and distribution costs limit the attractiveness of this option.

The digital library will maintain the URLs for the access centers. This will provide soldiers with easy access to available courseware. The integration of Web, database, and video resources will allow content creators to create, store, and share the various multimedia components.

## Future

The most significant recent advancements have been in Web-based training tools such as Java, scripting languages, dynamic HyperText Markup Language, cascading style sheets, streaming media, and database integration. These technologies have closed the gap between Web-based products and executables created with multimedia authoring products. In fact, most current computer-based training packages export to a Web format. TRADOC is currently upgrading its software to incorporate the management of these new capabilities. Inevitably, the quality of the courseware will determine the success of the program.

TRADOC plans to field several access centers beginning this year. To meet the needs of Force XXI, the access centers must allow access to today's users, leverage existing technology, and yet, not succumb to technological obsolescence.

Lt. Col. Anthony Ruocco  
E-mail: [ruocco@exmail.usma.edu](mailto:ruocco@exmail.usma.edu)

Maj. Todd L. Smith  
E-mail: [dt4571@exmail.usma.edu](mailto:dt4571@exmail.usma.edu)



# Expanding the Focus of Software Process Improvement to Include Systems Engineering

Kent A. Johnson and Joe Dindo  
TeraQuest Metrics, Inc.

*There is a growing interest in improving the efficiency of systems engineering in organizations that develop software-intensive systems. A number of organizations have demonstrated that it is possible to increase software process capability through software process improvement programs. As a result, there is a heightened interest in improving the processes used in systems engineering of software-intensive systems using systems engineering process improvement programs. This article addresses some motivations for improving the systems engineering process, gives an overview of systems engineering process models, and identifies how to leverage an organization's successes in software process improvement.*

There is a long history of organizations performing impressive feats of systems engineering (SE). The growing complexity of software-intensive systems has made SE increasingly important for embedded systems and information systems.

Although there is no generally accepted single definition of SE, the pursuit of excellence in SE continues (see sidebar – Systems Engineering Definitions). Increased interest in systems engineering is evident from the following:

- The creation and growth of the International Council on Systems Engineering (INCOSE).<sup>1</sup>
- The development of systems engineering process models.
- The government supported integration efforts for use of systems engineering models.

This article is written for the benefit of an organization or enterprise that contains multiple groups, some with responsibility for SE activities and others with responsibility for software engineering activities.

For the most part, the business of these organizations is oriented to the development of one or more software-intensive products such as printers, cellular telephones, automobiles, and weapons. The organization typically has a product development focus with product-focused groups that work with one or more other groups such as component engineering, software de-

velopment, marketing, sales, manufacturing, and service. The organization executes a product development lifecycle from which the various versions of the product are developed and released. It is recognized that many of these concepts also are applicable to information systems development.

Information systems development has a similar set of considerations, though the domains being brought together as a system might be different. The new information system may require development or acquisition of a computing platform, development or acquisition of software, definition of new processes, creation of training, hiring of a new team of workers, and establishing a help desk. Integrating all

of these disciplines around the creation of the new system requires many of the same processes as the systems engineering of product-focused groups. For simplicity and clarity, this article deals with product-focused organizations.

## Systems Engineering Process Models

Since 1992, several groups have developed SE process models. These groups include representatives from industry and government organizations with an established record of accomplishment in systems engineering of large complex systems. Table 1 gives a summary of three of the models that resulted from this work.

## Systems Engineering Definitions

### Systems Engineering – Definition 1

“Systems Engineering is the selective application of scientific and engineering efforts to

Transform operational need into a description of the system configuration that best satisfies operational need according to measures of effectiveness. Integrate related technical parameters and ensure compatibility of all physical, functional, and technical program interfaces in a manner that optimizes total system definition and design.

Integrate efforts of all engineering disciplines and specialties into total engineering effort.”<sup>5</sup> – *Systems Engineering Capability Maturity Model, Version 1.1.*

### Systems Engineering – Definition 2

“Systems Engineering is an interdisciplinary approach and means to enable the realization of successful systems.”<sup>6</sup> – *Draft version 0.5 of the Systems Engineering Capability Model EIA 731-1.*

Close examination of the Systems Engineering Capability Maturity Model<sup>SM</sup> (SE-CMM<sup>®</sup>)<sup>2</sup> [1], Systems Engineering Capability Assessment Model (SECAM) [2], and Systems Engineering Capability Model (SECM) [3] shows they have

- The same (continuous) architecture.
- Many of the same authors.
- Similar process or focus areas divided into categories for technical, management, and organizational elements [4].

To illustrate some of the attributes of an SE process model, Table 2 lists the 18 process areas of the SE-CMM. The table is divided into three categories: Engineering, Project, and Organizational. Process areas (or focus areas) of the SECAM and SECM are nearly identical; however, for consistency, the SE-CMM process names will be used throughout the remainder of this article.

The architecture of the SE-CMM is continuous, as is the architecture of the SECAM and the proposed architecture of the SECM models. The CMM for software uses a staged architecture.

### Staged Architecture Models

Staged models provide guidance to organizations on the order of improvement activities they should undertake based on (key) process areas at each stage or maturity level. Performing the practices in the appropriate process area at a given level will help stabilize projects and allow the execution of further improvement activities. Because all stages contain a collection of process areas on which to focus current activities, incremental improvement is supported in each maturity level or stage.

Each stage provides the foundation for the next stage, which promotes the “crawl before you walk” approach. However, some organizations will decide, for business or cultural reasons, to address certain process areas earlier than defined by the stages. For example, users of the CMM for software often address some Level 3 key process areas (KPAs) while the organi-

Systems Engineering Capability Maturity Model (SE-CMM)		
Developer: Enterprise Process Improvement Collaboration. Members from Lockheed Martin, SECAT LLC, Hughes, Department of Defense, Federal Aviation Administration, Software Productivity Consortium, Software Engineering Institute (SEI), Texas Instruments, National Institute of Standards and Technology, and others. (Released as SEI documents.)		
Architecture: Continuous	Elements: 18 Process Areas	Categories: Engineering, Project, and Organization
Current Version: SE-CMM, Version 1.1 published November 1995	Availability: No charge on the Web. See Frameworks Quagmire <a href="http://www.software.org/quagmire">http://www.software.org/quagmire</a>	Assessment Method: SE-CMM Appraisal Method, Version 1.1. March 1996
Status: The model is used by organizations around the world with third-party appraisals performed by organizations including TeraQuest, SECAT, and the Software Productivity Consortium.		
Systems Engineering Capability Assessment Model (SECAM)		
Developer: Capability Assessment Working Group of the International Council on Systems Engineering (INCOSE) with initial members from Grumman, Hughes, CSC, Loral, and Software Productivity Consortium (149 contributors through Version 1.50).		
Architecture: Continuous	Elements: 19 Key Focus Areas	Categories: Systems Engineering, Management, and Organization
Current Version: SECAM, Version 1.5 published July 1996.	Availability: Information available at <a href="http://www.incose.org">www.incose.org</a>	Assessment Method: Systems Engineering Capability Assessment Method
Status: Assessments being performed by INCOSE members.		
Systems Engineering Capability Model (SECM)		
Developer: Electronics Industries Association (EIA) G-47 Systems Engineering Committee. Project members include EPIC, INCOSE, and EIA representatives. This is a merger of the SE-CMM and the SECAM intended to replace the two with a single model that will serve as an International Organization for Standardization standard.		
Architecture: Continuous	Elements: 19 Focus Areas	Categories: Technical, Management, and Environment
Current Version: EIA IS 731 was presented for ballot distribution April 20, 1998. Could be published as an interim standard as early as July 1998.	Availability: As of this writing, not publicly available.	Assessment Method: To be included in the interim standard.
Status: Under development. Interim standard designation was selected because work on CMMI is expected to require SECM modification within a year.		

Table 1. *SE process models.*

zation is working on Level 2. Typically, an organization establishes a Software Engineering Process Group and begins work on the Organization Process Definition and Organization Process Focus (Level 3) KPAs. Another example is an organization that implements defect analysis (from the Level 5 KPA Defect Prevention) before establishing Level 2 processes. There are many cases where these early implementation approaches have led to failure or slow progress. This is often the case because foundation processes

such as project planning and tracking are missing, or because organizations lack focus from trying to incorporate too much change at one time.

### Continuous Architecture Models

Continuous models provide more flexibility in defining process improvement programs. These models recognize that individual process areas are performed at distinct capability or maturity levels. Based on the institutionalization of the base practices of that process area, a continuous model

Engineering Process Areas	Project Process Areas	Organizational Process Areas
1. Analyze Candidate Solutions	8. Ensure Quality	13. Define Organization's Systems Engineering Process
2. Derive and Allocate Requirements	9. Manage Configurations	14. Improve Organization's Systems Engineering Process
3. Evolve System Architecture	10. Manage Risks	15. Manage Product Line Evolution
4. Integrate Disciplines	11. Monitor and Control Technical Effort	16. Manage Systems Engineering Support Environment
5. Integrate System	12. Plan Technical Effort	17. Provide Ongoing Knowledge and Skills
6. Understand Customer Needs and Expectations		18. Coordinate with Suppliers
7. Verify and Validate System		

Table 2. *Process areas of the CMM.*

assessment provides a profile with specific maturity levels for each process area.

In the SE-CMM, the SE activities performed for a process area are structured as base practices within each process area. For each level of the process area, exemplary generic practices apply. For example, the generic practices for each level are:

- Level 1 – perform the process.
- Level 2 – document the process.
- Level 3 – tailor the standard practice.
- Level 4 – determine process capability.
- Level 5 – continuously improve the standard process.

To use these models for process improvement, organizations must perform an analysis of how the various processes address their needs. This takes advantage of the inherent flexibility of the model. Such an exercise also provides an opportunity for the organization to gain consensus on the sequence of appropriate organization-wide improvement activities.

Considering their understanding of staged models, a number of organizations with a significant investment in software CMM-based software process improvement (SPI) have sought an SE process model with a staged architecture. Having a staged SE model makes it easy to describe the model content to people in the organization and eases the integration of SE improvement

activities with current software process improvements.

Table 3 shows an SE staged model developed by mapping the SE-CMM process areas and generic practices onto four levels. This model was developed by the Process-Oriented Systems Engineering (POSE) project. The POSE project is a process improvement experiment that is part of the

European Systems and Software Initiative. The POSE project is managed by Thomson-CSF with additional members from TeraQuest Metrics, Inc. and the European Software Institute.

Corporations with multiple organizations or multiple business units can use such a model to support a synergistic approach to SE process improvement across the corporation. A review of process areas and generic practices associated with each level reveals that staging is roughly equivalent to the CMM for software.

This example SE model is incremental, with each stage building on the next. That is, to perform at Stage Two, an organization must perform the Level 2 generic practices listed in the left column for all eight of the process areas listed in the right column. To perform at Stage Three, an organization must perform the Level 2 and Level 3 generic practices for the Stage Two and Three process areas. Stage Four organizations perform Levels 2, 3, and 4 generic practices for

Table 3. *POSE Systems Engineering Staged Model (Version 2.0, May 1998).*

	Generic Practices	Process Areas
Stage Two	<i>Level 2 – Planned and Tracked</i> <ul style="list-style-type: none"> <li>• Planning performance</li> <li>• Disciplined performance</li> <li>• Verifying performance</li> <li>• Tracking performance</li> </ul>	Understand Customer Needs and Expectations (PA 6) Derive and Allocate Requirements (PA 2) Coordinate with Suppliers (PA 18) Plan Technical Effort (PA 12) Monitor and Control Technical Effort (PA 11) Manage Configurations (PA 9) Establish Quality Assurance (PA 8 – split) Control Contract (additional PA)
Stage Three	<i>Level 3 – Well Defined</i> <ul style="list-style-type: none"> <li>• Defining a standard process</li> <li>• Performing the standard process</li> </ul>	Define Organization's SE Process (PA 13) Provide Ongoing Knowledge and Skills (PA 17) Evolve System Architecture (PA 3) Integrate System (PA 5) Verify and Validate System (PA 7) Integrate Disciplines (PA 4) Adopt Product Line (PA 15 – split) Manage Risks (PA 10)
Stage Four	<i>Level 4 – Quantitatively Controlled</i> <ul style="list-style-type: none"> <li>• Establishing measurable quality goals</li> <li>• Objectively managing performance</li> </ul>	Ensure Quality (PA 8 – split) Manage SE Support Environment (PA 16)
Stage Five	<i>Level 5 – Continuously Improving</i> <ul style="list-style-type: none"> <li>• Improving organizational capability</li> <li>• Improving process effectiveness</li> </ul>	Improve Organization's SE Process (PA 14) Manage Product Line Evolution (PA 15 – split)

all process areas through Stage Four. Finally, Stage Five organizations must perform all generic practices for all process areas.

In two process areas, it has been necessary to split the process area across stages. An example of a split process area is Ensure Quality, which has been split into two process areas: Establish Quality Assurance at Stage Two and Ensure Quality at Stage 4. The POSE project also chose to remove the Analyze Candidate Solutions (PA 1) process area and add the Control Contract process area.

### Process Mismatch – Why Should You Care?

Organizations cannot perform to their full potential with a mismatch in the capability of the processes used by different development groups within the organization. A process mismatch exists when the software and systems processes operate at different levels of

maturity. Consequently, organizations experience the following:

- Difficulty in communication and commitments between project groups.
- Inability to effect improvements to their overall processes.
- Overall performance below the capability of the individual software or systems processes when considered on their own merit.

This situation becomes more evident as the number of organizations making significant progress in SPI increases. In these organizations, software processes are often at a higher maturity level than the SE processes. This is not an indication of the ability or professionalism of the engineers, but an indication of unaligned processes. There also are cases of higher maturity level systems groups that have interface mismatch problems with lower-level software groups; however, not much data exists since the number

of systems assessments remains relatively low.

Given the cost associated with product development and process improvement activities, it is prudent to promptly address these mismatches in capability. For example, an organization performing at CMM Level 1 needs to focus on the mechanisms by which project commitments are made and kept. They need to make cooperative commitments with the SE organization. For the development of the product, in such a case, both the software organization and the SE organization need project plans with adequate visibility into the market commitments and into their respective work. No other technical advances in the SE organization can make up for a mismatch in the basic commitment process.

The following sections are divided by software CMM maturity levels, and they discuss what the software development group needs from the SE group for the organization to best exploit the SPI investment. This involves leveraging the strengths of the systems engineering group in a collaborative manner so that the organization meets its overall goals.

### Level 2

A CMM Level 2 organization supports the basic commitment process. Requirements and schedules from and with SE are key to this success. Typically, software commitments are made between software project managers and agents of the product development groups such as program managers and systems engineers. The software organization needs the following from these agents

- Well-defined requirements (particularly those allocated to software).
- Negotiable schedules.
- A stable physical architecture.

### Level 3

A CMM Level 3 organization supports the required infrastructure for using a defined process. The essential element is coordination of all activities between the multiple disciplines and groups collaborating to build a prod-

Table 4. Contributions of systems engineering to process capability.

Level	Organizational Behavior	System Engineering Contribution	SE-CMM Process Areas
2	Commitments	• Well-Defined Requirements	• Derive and Allocate Requirements (Pa 2) • Understand Customer Needs and Expectations (PA 6)
		• Negotiable Schedules	• Manage Risks (PA 10) • Monitor and Control Technical Effort (PA 11) • Plan Technical Effort (PA 12)
		• Stable Physical Architecture	• Derive and Allocate Requirements (PA 2) • Manage Configurations (PA 9)
3	Organization-Wide Standardization	• Well-Understood System Architecture	• Evolve System Architecture (PA 3) • Integrate System (PA 5) • Verify and Validate System (PA 7) • Manage Configurations (PA 9)
		• Well-Defined Organizational Processes	• Define Organization's Systems Engineering Process (PA 13) • Improve Organization's Systems Engineering Process (PA 14)
		• Clear Organizational Interfaces	• Integrate Disciplines (PA 4)
4	Quantitative Understanding	• Well-Managed Product Line Evolution	• Manage Product Line Evolution (PA 15)
		• Well-Managed Engineering Support Environment	• Manage Systems Engineering Support Environment (PA 16)
5	Continuous Improvement	• Organization-Wide Quality Focus	• Ensure Quality (PA 8)
		• Continuous Improvement of Defined Processes	• Improve Organization's Systems Engineering Process (PA 14)

Organizational Consideration			
Approaches	Existing Software Process Improvement Program	Focus of Engineering Organization	Urgency of Change
1 Use Existing SE Model	(+) May help get buy-in from Systems Groups (-) May not exploit the existing program's work	(+) If systems focused (-) If software focused	(+) Project Process Areas provide triage (+) Available today
2 Add System Process Areas to CMM for Software	(+) If at software Level 3, can add Engineering Process Areas just like Software Product Engineering KPA	(+) If software focused (-) If system focused	(-) Will take some time and effort to define merger
3 Add Software KPAs to SE Model	(-) May not make best use of existing Software Process Improvement Infrastructure	(+) If systems focused (-) If software focused	(-) Will take some time and effort to define merger
4 Wait for SECM or CMMI to complete	(+) No impact on current program (-) No impact on current program	(+) CMMI should provide for a merged focus	(-) Likely to be some time in coming

Table 5. Approach consideration matrix.

uct. The product development groups need to ensure that they all have the following with which to work.

- A well-understood system architecture.
- Well-defined processes.
- Clear organizational interfaces.

Systems architecture is defined as “The fundamental and unifying system structure defined in terms of system elements, interfaces, processes, constraints, and behaviors.”<sup>3</sup>

#### Level 4

A CMM Level 4 organization provides a clear and quantitative definition of its products and processes. Organizations moving into Level 4 begin to see increased benefit from reuse programs as the products produced by the organization have a higher recognized quality. The product development groups need to provide the following:

- A well-managed product line evolution.
- A well-managed engineering support environment.

Product-line engineering is defined as “The engineering of a family of products that is developed using a domain analysis approach and share the same system architecture.”<sup>4</sup>

#### Level 5

A CMM Level 5 organization performs continuous process and product improvement. Through defect analysis, the organization identifies root causes and eliminates them at the source. The product development groups need to provide the following:

- An organization-wide quality focus.
- Continuous improvement of defined processes.

Table 4 summarizes the above and provides a cross-reference to the specific SE-CMM process areas that

support each software CMM level as described above.

### Using the SE Capability Models

After reviewing the systems engineering process models and reading about the CMM integration activities (see sidebar – CMM Integration Project), many organizations have asked the following questions.

- How can we use these various models?
- What should we do next?

The answer to the first question is familiar to those who have been performing SPI: Any one of the SE models can be used to identify process assets, perform systems engineering process assessments, and identify and exploit the organization's best practices. The key—prompted by the number of models as well as the changes in these SE models—lies in the second question. The answer to the second question is to select one of the following alternative approaches (Table 5).

Approach 1 – Use an existing systems model (SE-CMM or SECAM).

Approach 2 – Add System Process Areas to the software CMM.

Approach 3 – Add software KPAs to a systems model.

Approach 4 – Wait until one of the integration efforts is completed.

## CMM Integration Project

The CMM integration project [5] was initiated in response to organizations using multiple process models that need a consistent process model to apply to improvement activities. The project is sponsored by the U.S. DoD, Office of the Secretary of Defense for Acquisition and Technology at the Software Engineering Institute. A product of the integration project is the Common CMM Framework (CCF), which allows for both continuous and staged model representations (CCF Draft E). The CCF is represented by a set of standard requirements for all CMMs. Little public information is currently available; however, the CMMI Web page at [www.sei.cmu.edu](http://www.sei.cmu.edu) promises additional information on an ongoing basis. The stated project goals are to

- Enable accelerated release of a software CMM equivalent to Version 2.0, Draft C under CCF, which is ISO 15504 compatible.
- Provide complete product suite—complete model plus assessment and training materials.
- Provide opportunities for all industry and government organizations to participate.

To select between the alternatives, the organization needs to consider the current improvement programs, the engineering focus of the organization and its products, and the level of need or urgency for change.

### Current Improvement Programs

The aim is to exploit any successes, momentum, and products of existing SPI programs. The organization should consider the following questions.

- Is there an active SPI program?
- Is it well accepted by the organization?
- Is the program making progress?
- Is the organization now working to attain Level 3 or above?

If the answer to some or all of these questions is “yes,” there are strong reasons to leverage this momentum. On the other hand, if the ongoing SPI program is slow moving, not accepted, or regressing, it may be wiser to start over with a fresh approach.

### Organization’s Engineering Focus

The organization’s engineering focus or cultural perspective will have an impact on how best to approach the problem. The organization should consider the following questions.

- Is the organization primarily in the business of producing systems or software?
- What is the focus of most new development?
- Where is the locus of control?

If the business is a software business inside another apparent business, it is wise to focus on the software CMM.

### Urgency for Change

A pressing need that is visible to the organization motivates action. The organization should consider the following questions.

- Is the organization facing near death?
- Is there a major program in trouble?

Organizations that are in significant trouble or almost going out of business are sometimes more willing

to proceed with improvement efforts than successful organizations. Such organizations can use the information from the process areas to address immediate project issues.

Table 5 shows the positive and negative impact of these organizational considerations based upon the four approaches. For example, the first approach (Use Existing SE Model) is an effective choice for organizations that are seeking buy-in from their systems groups, need quick improvement of their overall project management capabilities, and need something immediately. The buy-in is easier with these models since they were developed by systems engineers for systems engineering. As such, these models can be used as they exist today; what is needed is an integration of the chosen SE model with the organization’s current SPI efforts, either through an organizational integration effort or through an external integration project.

### Recommendations

In spite of the various SE models and the development activities surrounding them, we strongly suggest that you do not wait. Based on your situation, select any of the first three approaches suggested above and get started. Although work is proceeding on the integration of the systems process models, these integration activities will take some time before a model emerges that meets the ever-growing set of needs. Further, an SE process improvement (SEPI) program based on one of the existing models should require minimal effort to migrate to a new model when it becomes available. Remember that all the models are developed from the same basic source materials. The benefit of progress from an active SEPI program should outweigh any risk of rework that results from a new SE model. The risk can be minimized by staying current with model developments and making progress in your SEPI program. ♦

### About the Authors



**Kent A. Johnson** is the director of systems engineering for TeraQuest Metrics, Inc., where he has helped companies in five countries to improve their systems and software processes through systems and software assessments, training, and process-specific consulting. He is a former project manager of the Process and Methods Program at the Software Productivity Consortium (SPC). While at the SPC, he co-wrote the software and system development method, as well as led the team that helped create it, which is used by over 800 engineers in the development of the F-22 Advanced Tactical Fighter. He is also a co-author of the Spring-Verlag book, *Ada 95 Quality and Style*.

TeraQuest Metrics, Inc.  
10812 Monticello Drive  
Great Falls, VA 22066-4224  
Voice: 703-404-9769  
Fax: 512-219-0587  
E-mail: johnson@teraquest.com  
Internet: <http://www.teraquest.com>



**Joe Dindo** is a senior associate at TeraQuest, where he provides training and process improvement coaching to clients working to improve their systems and software development capabilities. He has over 10 years experience working in all phases of the systems engineering lifecycle, its supporting processes, and process improvement activities. His engineering experience encompasses a wide range of roles using multiple lifecycle models and methods. His professional responsibilities have included managing small- and medium-sized projects and providing leadership to project and middle management teams.

TeraQuest Metrics, Inc.  
6772 Locust Drive  
Troy, MI 48098  
Voice: 248-879-2947  
Fax: 512-219-0587  
E-mail: dindo@teraquest.com  
Internet: <http://www.teraquest.com>

### References

1. Bate, Roger, et al., *A Systems Engineering Capability Maturity Model*, Version 1.1, Handbook SECM-95-01, Software



This article can be found in its entirety on the Software Technology Support Center Web site at <http://www.stsc.hill.af.mil/CrossTalk/crostalk.html>. Go to the "Web Addition" section of the table of contents.

## DO-178B: Software Considerations in Airborne Systems and Equipment Certification

Leslie A. Schad  
Boeing Commercial Airplane Group

*This is a practitioner's discussion of the evolution of the current practice and application of RTCA/DO-178B[1] for software approval in the commercial world. The objectives include developing and providing the data for development of educational material, providing the rationale behind the guidance for people new to the commercial certification environment, and clarification of the intent and application of DO-178B. The derivation of the software approval guidelines from the Federal Aviation Regulations to DO-178B is discussed to clarify its relationship to the government regulations. An explanation of the Designated Engineering Representative system is also provided along with a discussion of the safety process to describe the environment in which DO-178B is used. The evolution of the avionics industry that eventually led to DO-178B is included as part of the background behind the rationale of DO-178B. The key aspects of each version, from the original version to DO-178B, provide insight to the rationale for the inclusion and further development of the content. In addition, there are special considerations in using DO-178B concerning its current guidance for systems and highlights of the problem areas for those from a military culture. As the industry moves to use of commercial-off-the-shelf components, the incentive is greater to reconcile the difference between military standards and commercial standards. Trustworthiness of software is an absolute concept independent of the verification process used. This article explores the differences and similarities between DO-178B and MIL-STD-498 affecting the software development process.*

- Engineering Institute, Carnegie Mellon University, Pittsburgh, Pa., 1995.
2. SECAM: *Systems Engineering Capability Assessment Model*, Version 1.5, INCOSE, 1996.
  3. SECM: Not yet released; planned as EIA IS 731, Electronics Industries Alliance, Alexandria, Va.
  4. Sheard, Sarah A. and Jerome G. Lake, "Systems Engineering Standards and Models Compared," Software Productivity Consortium, Herndon, Va. (<http://www.software.org/pub/papers>), April 15, 1998.
  5. Schaeffer, Mark D., "Capability Maturity Model Process Improvement," *CROSSTALK*, Software Technology Support Center, Hill Air Force Base, Utah, May 1998.
  3. Definition approved by the Systems Architecture Working Group of INCOSE at INCOSE '96, Boston, Mass.
  4. Definition approved by the Systems Architecture Working Group of INCOSE at INCOSE '96, Boston, Mass.
  5. Army Field Manual, pp. 770-778.
  6. Definition approved by the International Council on Systems Engineering at the January 1996 workshop.

### Notes

1. INCOSE can be found at <http://www.incose.org>.
2. Capability Maturity Model is a service mark of Carnegie Mellon University. CMM is registered with the U.S. Patent and Trademark Office.



### New Crosstalk Phone Numbers

Some of Crosstalk's phone numbers have changed. Please use the phone numbers in the masthead on page 31.

# “Honey, I Forgot the Users!”

Lori Pajerek  
Lockheed Martin

*We have all heard some human factors horror stories, and as consumers, most of us have had numerous personal experiences with products that are difficult, dangerous, or annoying to use. Their designers did not seem to have humans in mind. Poor human factors designs are the result of oversight or ignorance. The designers did not do enough work to ensure that the demands placed on the humans are not onerous. To address this shortcoming, all types of design engineers should learn to apply a systems perspective to their part of a system.*

Which is better—mouse or keyboard? For example, since the introduction of the computer mouse, numerous interactive computer operations are now “point and click” even though the mouse is the slowest interface for performing many operations. It takes time to move the mouse into position and click it, especially when it must point to a relatively small area on the screen (such as a “soft button”). Although older technology than a mouse, using a function key when possible is not only much quicker, it also puts less wear and tear on your body.

This became apparent to me last winter when I used a new software tool intensively for extended periods. A certain operation required three mouse clicks every time: one to select the item on the screen to be acted upon, one to click the button bar to indicate the desired operation, and one to close a dialog box that asked, “Are you sure?” The three mouse clicks were in widely diverse areas of the screen, each required precise placement of the cursor in a tiny spot on the screen, and the computer took a lot of time to process each click.

Since I was performing this operation on many hundreds of individual items during the course of a day, I could not help but notice how slow the “mouse clicking” process was. Also, my fingers, wrist, arm, and shoulder ached at the end of the day. It was much more time-consuming than performing the same operation on a competing vendor’s tool, which took only one mouse click. Additionally, for most operations, the other tool often

gave me a choice between using the mouse or a function key. Having found that my wrist and arm got more tired and painful from repeatedly using the mouse, I prefer the function key when given the choice. Chalk one up for the competition.

This story illustrates how my personal experience led me to conclude the fewer mouse clicks, the better. But this is nothing more than a subjective opinion. The world loves mice. I may be the only person in the world complaining about too many mouse clicks. However, I recently saw some hard data from an internal IBM human factors study that empirically demonstrates that the mouse is the slowest of several common data entry devices. Yet, most software designers do not know this—they are often not aware that such studies exist. Because the mouse is newer technology than the keyboard, many of today’s designers—who grew up using a mouse—assume it is better. Even though graphical user interfaces (GUIs) invite the use of a mouse, it may not always be the optimal choice.

Human factors is an area where applying the systems perspective to engineering is key. The systems perspective requires looking at all parts of a system throughout all phases of system development and deployment and questioning how a system will be used. For human factors considerations, this means performing operational scenarios early in the system development to determine human interaction with the system, specifying human factors requirements, reviewing designs for human factors impacts, and modeling

the user interface in mock-ups. This activity should continue through system verification to validate assumptions, scenarios, and data. At this point in development, it is usually too late and too expensive to reverse any adverse design decisions—unless they seriously detract from the usability of the system.

The vendors of the “slow” tool described above may have developed operational scenarios, but unless they tested them with users under loaded network conditions, they would not have discovered what the implications were for the human using the system to perform the operation hundreds of times in a row. Nonetheless, it is a highly likely operational scenario for that particular tool.

## The Personnel Subsystem

Recently, I learned a term that I had never heard, but one that I immediately liked, the “Personnel Subsystem.” Engineers have become better at remembering the human in the system. The user interface on any system usually has ergonomics requirements applied to it, and conceptual diagrams of systems often have a stick figure or clip art of a person representing the user(s). We know they are there, and we acknowledge their existence to some degree.

Specifications usually contain a section called “Personnel and Training”—it was required by MIL-STD-490. Generally, there are some requirements in the specifications; usually high-level statements that specify education, training, and skills for different users (operators,

maintainers, etc.). But while I have often seen an architectural element called "Training" in system architectures, I have yet to see a personnel subsystem, beyond the stick figure or clip art variety. It is the *forgotten subsystem*.

We know there are requirements that should be allocated to a personnel subsystem, yet we design this subsystem by default. Every time you specify or design a human interface, you are also implicitly specifying or designing the expected characteristics and behavior of the human half of the interface. Humans must perform to this implicit specification, but there is never a requirements review or design review of the personnel subsystem. Unless your system is never touched by human hands (a rarity), you have a personnel subsystem, whether you identify it explicitly or not. Just because you do not have to manufacture it does not mean it does not exist; it is a real part of the system integration picture.

By including a personnel subsystem in your system architecture, you are more likely to achieve the objective of looking at all parts of the system through all phases, including the people part of the system. Though we sometimes assign a few requirements to users, we often do not follow through. How do we know if we have "specified" a user with inappropriate or inconsistent requirements? You treat the personnel subsystem like other subsystems—review the requirements and verify the design.

It also is important to remember that consideration for the personnel subsystem is not limited to physical ergonomics. There are other factors involved that relate to the users' mental processes. These range from straightforward things such as consistent menu design to more subtle intangibles such as impact on job satisfaction.

## Conclusion

Human factors is following a path that has been experienced in a number of other traditional engineering disciplines. Concepts like quality control, configuration management, and plug-and-play interfaces were all institutionalized in hardware engineering long before they were applied to software. Ergonomics has become a serious consideration for all types of hardware items. We have furniture, CRTs, keyboards, mice, wrist rests, and all kinds of other things that feature ergonomic design. In software, we now have windows and GUIs, but these features merely scratch the surface of what is possible. However, the systems perspective is what motivates someone to investigate the effects of the combined interaction of the ergonomic hardware and the user-friendly software with the personnel subsystem. What happens when a human sits for hours in the ergonomic chair, using the ergonomic mouse to click thousands of times on the software-generated GUI? All the pieces were designed to be kind to humans, but the integrated system still causes the user to go home in the evening with eye-strain and a sore neck. When designers think like users, the result is a system that is truly user friendly. ♦

## About the Author

**Lori Pajerek** is an advisory systems engineer in the Systems Engineering Technology department at Lockheed Martin Federal Systems in Owego, N. Y. With over 15 years experience in systems and software engineering for defense-related industries, her specific area of interest and expertise is requirements engineering and requirements management. Prior to joining Lockheed Martin, she was employed at Link Flight Simulation (now part of Raytheon), the world's leading manufacturer of flight simulation devices. She has a bachelor's degree in mathematical sciences from Binghamton University and is a member of the International Council on Systems Engineering.

Lockheed Martin Federal Systems  
1801 State Route 17C  
Mail Drop 0902  
Owego, NY 13827  
Voice: 607-751-6226  
Fax: 607-751-2008  
E-mail: lori.pajerek@lmco.com

## Systems Engineering at the STSC

The newest systems engineering technologies can be confusing—SECM, SECAM, SE-CMM, CMMI, 632, 731, 1220, 15288. The Software Technology Support Center can help you sort through the jumble. We partner with you to identify, evaluate, and adopt effective technologies that improve product quality and process efficiency and predictability. Our consultants work directly with you to tailor proven processes, methods, and tools to your organization's needs.

- **Process improvement:** assess, plan, implement, and lead improvement efforts that keep you competitive.
- **Technology evaluation:** identify, pare down, evaluate, and select what best meets your needs.
- **Technology adoption.** Understand, apply, and exploit technologies to improve quality, productivity, and customer satisfaction.

Although our information is free, we provide on-site briefings, training, workshops, assessments, planning, and guidance on a cost-recovery basis. The STSC specializes in

- Requirements definition, design, documentation, test, and reengineering.
- Acquisition, project, and configuration management.
- Assessments and metrics.

## Call us for your systems engineering needs.



Randall R. Wright  
Voice: 801-777-9732  
E-mail: wrightr@software.hill.af.mil

Russell Lee  
Voice: 801-775-5740  
E-mail: leeru@software.hill.af.mil

# Improvement Stages

Kerinia Cusick  
SECAT LLC

*Let us face it: A Capability Maturity Model (CMM) with a single number assessment rating system, such as the CMM for software, is easier to communicate and understand than a multinumber model, such as the Systems Engineering (SE) CMM. Improvement Stages are a way to organize the information in the SE-CMM to simplify the model. As an added benefit, Improvement Stages can be used to bridge the gap between the systems engineering and software CMMs.*

The SE-CMM is a tool designed to help organizations measure and improve their systems engineering processes. It is sometimes called a “continuous model,” which means the architecture is designed to provide the user with much flexibility and to loosely describe how companies should structure their improvement plans. This contrasts with the CMM for software, termed a “staged model,” which uses a more structured and prescriptive architecture that describes a clear sequence for improvement through its maturity levels.

The result of an assessment against a continuous model is a rating profile that gives you a different number for each assessed area, whereas the result of staged model assessment is a single number. Some companies appreciate the flexibility of continuous models, but many find them overly complex, leaving potential users confused and unable to develop an effective plan of attack for deploying the model within their own companies.

There are a number of staged and continuous models being used by industry. While the concepts in this article apply to other models, this article focuses strictly on the Systems Engineering and Software Capability Maturity Models.

## Systems Engineering CMM Description

The SE-CMM describes the essential systems engineering and management tasks that any organization needs to perform. These essential tasks are organized into logical groupings called *Process Areas* (all are listed in Figure 1.) The manner in which these essential systems engineering tasks are performed can range from completely ad hoc to continuously improved using statistical data. For each process area, this progression is broken into five primary steps, called *Capability Levels* in the SE-CMM, each of which lays the foundation for the next step.

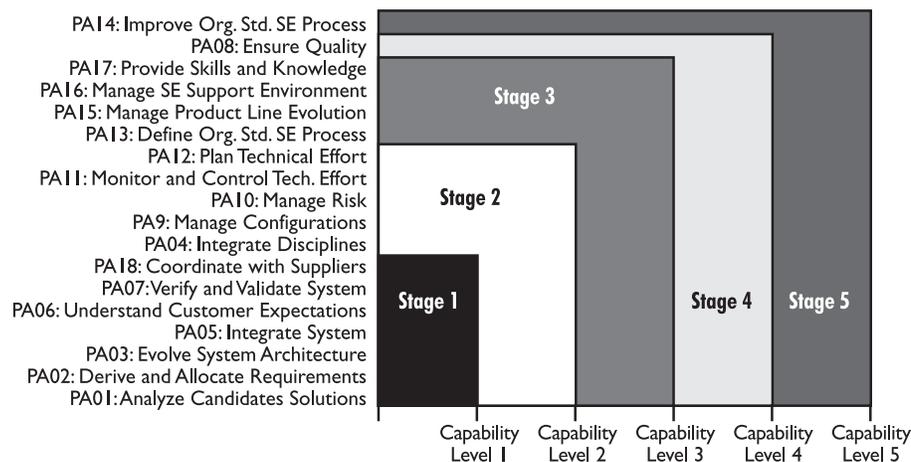
The SE-CMM architecture allows users to decide which systems engi-

neering tasks (process areas) are essential or most important to their line of work, then lets them decide how well they want to manage those essential tasks (at what Capability Level they want to perform each process area). Because there are 18 process areas in the model, the user has much autonomy but also has many decisions to make.

Many organizations prefer to set performance goals against Capability Maturity Models. Management may be familiar with the software CMM and make statements such as they “want to be Level 2 by the end of the year.” Although that statement has a clear meaning in the software CMM, its meaning in the SE-CMM is less than obvious. It is often interpreted at being capability Level 2 in all process areas, but does that make sense?

Most users do not realize that some SE-CMM process areas are far more difficult to accomplish than others. Some have a broader scope, require participation from all levels in the company, or are based on a detailed understanding of an organization's ability to develop a product. An analogy is comparing the SE-CMM process areas to educational classes. Some process areas are at the high school level, whereas others would be completed as part of a doctoral program. And although it may be fair to expect a high school student to get an “A” in a high school-level class, you cannot conclude that the same student is a failure for getting a “D” in a graduate-level class. Quite to the contrary, you would be proud that the student was passing the class. Like-

Figure 1. *Improvement Stages*



wise, it is unrealistic to expect that a company just starting on the road to continuously improved product development processes is going to achieve a capability Level 2 in all process areas.

## Improvement Stages

Close examination shows a link between the content of the SE-CMM process areas and the concepts embedded in the capability levels. Because

capability levels are designed to represent a gradual progression of improved management and processes, mapping the process areas to the concepts inherent in the capability levels provides

# Getting Beyond the Differences Using the Systems Engineering and Software Capability Maturity Models Together

If you are familiar with the Systems Engineering (SE) and the Software Capability Maturity Models (CMM), you know they have different architectures but much overlap and that the concepts behind the increasing level of process management and control (called capability level in the SE-CMM and maturity level in the software CMM) are nearly identical.

You may ask: "Why bother using two different models when they have about a 70 percent overlap?" In fact, some organizations do want to merge the software and SE-CMMs to better improve their presently separate efforts. The Federal Aviation Administration even created its own integrated Capability Maturity Model. On the other hand, some prefer to keep the models separate, but still need greater coordination between their separate improvement efforts.

No matter what approach you choose, it is important to overcome the differences in architecture. One solution is discussed in this article: "Improvement Stages," which allow you to more clearly see the commonality between the two CMMs. There is some discontinuity because there is no software CMM equivalent of capability Level 1 or Improvement Stage 1. But looking at Improvement Stage 2 through 5, you can see the similarities shown in the table.

Use Improvement Stages to coordinate SE and software improvement efforts. If your goal is maturity Level 3 in the software CMM, you should also try to achieve Improvement Stage 3 in the SE-CMM. If the software and SE improvement groups are both trying to get the organization to maturity Level 3 and Improvement Stage 3 respectively, they will have the added benefit of trying to achieve similar objectives and facing identical problems.

Should you be interested in creating a systems-software model for your own purposes, use the Improvement Stages to make the translation. If you choose to incorporate the SE content into the SW-CMM, pull the SE-specific process areas and capability level practices from the SE-CMM and slide them over to the SW-CMM, keeping them at the same level. If the SE-unique process area was at Improvement Stage 3, it should be at maturity Level 3 in the SW-CMM.

SW-CMM Level	SW-CMM Key Process Area Name	Equivalent SE-CMM Process Area Name	SE-CMM Improvement Stage
2	Requirements Management	(no equivalent)	
2	SW Project Planning	Plan Technical Effort	2
2	SW Project Tracking and Oversight	Monitor and Control Technical Effort	2
2	SW Quality Assurance	Ensure Quality (portion of PA only)	2
2	SW Configuration Management	Manage Configurations	2
2	SW Subcontract Management	Coordinate with Suppliers (started in Improvement Stage 1, but now performed at equivalent level)	2
3	Organizational Process Focus	(similar PA in SE-CMM, but not a clear equivalent)	
3	Organizational Process Definition	Define Organization's SE Process (portion of process area)	3
3	Training Program	Provide Ongoing Skills and Knowledge	3
3	Integrated SW Management	Define Organization's SE Process (portion of process area)	3
3	SW Product Engineering	(no equivalent)	
3	Intergroup Coordination	Integrate Disciplines (started in Improvement Stage 2, but now performed at equivalent level)	3
3	Peer Reviews	Defect review practice at Capability Level 3	3
4	SW Quality Management	Ensure Quality (portion of PA)	4
4	Quantitative Process Management	Ensure Quality (portion of PA)	4
5	Process Change Management	Improve Organization's SE Process	5
5	Technology Change Management	(no equivalent)	
5	Defect Prevention	Causal analysis practice at Capability Level 5	5

an excellent method of organizing the process areas by level of complexity. The following describes how the 18 areas were mapped to a five-stage “Improvement Stages” model.

- **Capability Level 1** – ad hoc performance. The primary focus is on getting the system out; few, if any processes are in place. Therefore, the process areas that address performing the systems engineering activities are mapped to this level.
- **Capability Level 2** – characterized by planning and tracking within projects. This level includes process areas that deal with project management.
- **Capability Level 3** – the key concept is development and use of organizational standards and achieving an aligned organization. Includes all of the process areas that discuss organizational-wide activities or the development of standards.
- **Capability Level 4** – characterized by statistical process control; process areas that discuss measuring process quality quantitatively are mapped to this level.
- **Capability Level 5** – primarily characterized by continuous improvement using statistical process control data; therefore, the process area that addresses improving the standard process maps to Level 5.

Figure 2 shows the result of mapping the SE-CMM process areas to the capability level concepts. To avoid confusion with other terminology, we call the result of this mapping *Improvement Stages*. Listed on the vertical axis are the 18 process areas in the SE-CMM. Across the horizontal axis are the capability levels. Each Improvement Stage is cumulative, adding on more process areas and capability levels. For example, Stage 2 requires performing all 12 process areas (from Analyze Candidate Solutions to Plan Technical Effort) at a capability Level

2. Stage 3 adds on another four process areas, making a total of 16 process areas that must be performed at a capability Level 3.

Unfortunately, since the authors of the SE-CMM did not have the staging concept in mind when writing the model, the concepts in some process areas span multiple maturity levels. The two problematic process areas are “Ensure Quality” and “Improve Organization’s Standard Systems Engineering Process,” which both have content that maps to lower capability levels. To avoid encouraging companies from trying to implement these process areas in a manner that does not make sense, they were placed at the higher capability level.

Improvement Stages arrange the processes areas by order of difficulty. I do not mean to imply that a company should put on blinders, not considering any of the process areas in Improvement Stage 3 until they have completely mastered the process areas in Stage 2. One company using this system has referred to Improvement Stages as a primary area of emphasis.

Think of Improvement Stages as a ski slope map. The SE-CMM process areas are similar to a map showing only the location of trails on a mountain. Viewing this map, the skier knows how many slopes there are and where they are but knows nothing of the level of difficulty. The Improvement Stages concept is similar to knowing which trails are appropriate for beginners, intermediates, and advanced skiers. You can always start skiing on the advanced slopes as a beginner, but the odds are good that you will break your neck; likewise, you can tackle a hard process area first, but the odds are good that you will not be able to achieve your expectations.

## Conclusions

Although the SE-CMM is an effective systems engineering process measure-

ment and improvement tool, it presents the users with a measure of flexibility that can almost be harmful if too little time is spent to understand the content of the model and the complexity of the individual practices within each of the process areas. Many managers are setting company-wide, single-number goals without understanding that the SE-CMM has a different architecture than the software CMM.

However, even when managers do understand the SE-CMM structure, they are often unsure how to interpret the results of an assessment. Many even look at the 18-number profile and immediately calculate an average, determining that their organization is, for example, a 2.4. Improvement Stages provide a method of using a more meaningful single number score. If your company wants to be “Level 2,” consider restating the goal to be “Improvement Stage 2.” ♦

## About the Author

**Kerinia Cusick**, a co-founder of SECAT LLC, is one of the authors of the Industrial Collaboration Systems Engineering Capability Maturity Model and an author of the Integrated Product Development Capability Maturity Model. She is an experienced CMM teacher, assessor, and process improvement leader. She started her career at Grumman Aerospace working on digital flight control systems for experimental and fighter aircraft. At Hughes Missile Systems and Hughes Space and Communications, she transferred to systems engineering, working projects ranging from commercial communication satellites to Space Defense Initiative conceptual design studies. She has a master’s degree in systems management from the University of Southern California and a bachelor’s degree in mechanical engineering from Drexel University.

14742 Beach Blvd., #405  
La Mirada, CA 90638-4217  
Voice: 714-449-0423  
E-mail: kcusick@secat.com

# Coming Events

## SIGAda '98 (Formerly Tri-Ada): "Ada in Context"

**Dates:** Nov. 8-12, 1998

**Location:** Washington, D.C.

**Sponsors:** Association for Computing Machinery (ACM) Special Interest Group (SIG) on Ada in cooperation with ACM SIG on Applied Computing, SIG on Biomedical Computing, SIG on Computers and Society, SIG on Computer Science Education, SIG on Programming Languages, SIG on Software Engineering, and Ada-Europe.

**Subject:** This largest Ada-focused conference will be organized to attract participants from all segments of the software engineering community.

**Contact:** <http://www.acm.org/sigada/conf/sa98>

## Second International Quality Week Europe '98

**Theme:** Europe and Year 2000:  
The Industrial Impact

**Dates:** Nov. 9-13, 1998

**Location:** Brussels, Belgium

**Sponsors:** Association for Computing Machinery, European Software Institute, European Software Systems Institute, Software Research, Inc.

**Subject:** Focus on advances in software test technology, quality control, risk management, software safety, and test automation.

**Contact:** Rita Bral, conference director, Software Research Institute, 325 Third Street, Fourth Floor, San Francisco, CA 94107-1997

**Voice:** 415-957-1441

**Fax:** 415-957-0730

**E-mail:** [bral@soft.com](mailto:bral@soft.com)

**Internet:** <http://www.soft.com/QualWeek/QWE98>

## Metrics 1998: The Fifth International Symposium on Software Metrics

**Dates:** Nov. 20-21, 1998

**Location:** Hyatt Regency Hotel,  
Bethesda, Md.

**Sponsor:** IEEE Computer Society  
Technical Council on Software  
Engineering

**Subject:** This symposium presents the latest results in software engineering measurement as well as several state-of-the-art reports and panel presentations.

**Contact:** Mark Zerkowitz, program chairman, University of Maryland, Dept. of Computer Science, College Park, MD 20742

**Voice:** 301-405-2690

**Fax:** 301-405-3691

**E-mail:** [mvz@cs.umd.edu](mailto:mvz@cs.umd.edu)

**Internet:** <http://aaron.cs.umd.edu/metrics98>

## 18th Annual International Software Testing Conference and EXPO

**Theme:** "Testing for the Year 2000  
and Beyond"

**Dates:** Conference, Nov. 16-20,  
1998; EXPO, Nov. 19, 1998

**Location:** Orlando, Fla.

**Subject:** Conference tracks will include real-world practitioner presentations on year 2000 testing techniques, year 2000 success stories, testing new technologies, test management, testing techniques.

**Sponsor:** Quality Assurance Institute

**Voice:** 407-363-1111

**Fax:** 407-363-1112

**E-mail:** [lisag@qaiusa.com](mailto:lisag@qaiusa.com)

## Federal Imaging

**Dates:** Dec. 1-3, 1998

**Location:** Washington, D.C.

**Subject:** The inaugural edition of FEDnet'98, FOSE's Internet and Government Electronic Commerce and exhibition will fill a

niche by serving the special Internet and Intranet needs of the federal government.

**Voice:** 44-181-910-7878

**Fax:** 44-181-910-7813

**E-mail:** [inquiry@federalalimaging.reedexpo.com](mailto:inquiry@federalalimaging.reedexpo.com)

**Internet:** <http://www.reedexpo.com/factsheets/1635.html>

## FOSE: America's Integrated Information Technology Exposition

**Dates:** March 1, 1999

**Location:** Washington, D.C.

**Subject:** FOSE is the largest Integrated Information Technology trade event serving all branches of government. FOSE offers the full spectrum of information technology solutions to the largest computer market in the world.

**E-mail:** [inquiry@fose.reedexpo.com](mailto:inquiry@fose.reedexpo.com)

**Internet:** <http://fose.reedexpo.com>

## 11th Software Engineering Process Group Conference: SEPG '99

**Dates:** March 8-11, 1999

**Location:** Atlanta, Ga.

**Subject:** This four-day event brings together international representatives from government, industry, and academia for a truly global perspective on software process improvement.

**Sponsor:** Software Engineering Institute

**Voice:** 412-268-3007

**Fax:** 412-268-5758

**E-mail:** [sepg@sei.cmu.edu](mailto:sepg@sei.cmu.edu)



# Overcoming System Design Challenges

## The Creation of a System Description Language

Ingmar Ögren  
TofS AB

*It is difficult for system developers to create designs that represent the system in a manner that is understandable, detailed, and useful enough to everyone who must give input to the design. Yet, no matter how well this obstacle is dealt with, it is still difficult to find and fix design flaws in written system descriptions. This article describes Odel (Object Design Language), a readable, "executable" system description language that can be mastered by hardware and software developers in a couple of hours and that can easily be explained to end users. Not only does Odel increase understanding among all players but also combined hardware, software, and user descriptions written in Odel can be electronically analyzed and debugged to allow designers to spot incompleteness, flaws, and inconsistencies long before the design is implemented.*

**S**ystems engineering requires that developers work at the system level, which means they must understand the requirements of and interaction between hardware, software, and end users. End users, who usually have little experience with systems development, need a detailed understanding of how they will be required to jointly operate with software and hardware modules to complete their missions; without this understanding, they may not be able to provide the quality of input needed during the design process.

As a result, systems engineers' problems largely concern barriers to understanding, as they must design and describe complex systems in a way that can be readily understood by software designers, hardware designers, and end users. To create a system description that is correct and can be analyzed, it helps if the description is both understandable and reasonably formal.

### Evolution of Odel

The Ada programming language was introduced to the Swedish defense community about 11 years ago. To popularize Ada, the Swedish government contracted with Swedish contractor Sypro to create a simple Ada-based pseudo language to introduce Ada's semantics and syntax. The language, named "Adel" (Ada-based Design Language), was designed with simplifications of Ada, including the following:

- Tasking was simplified into a "concur" construct where a set of concurrent procedures are listed between the new reserved words "concur" and "end concur."
- Type and interface management was informal.
- The control structures from Ada were retained with some minor simplifications.

To support the language, an analyzer named Adela (Adel Analyzer) was developed to help write correct Adel. The Adel language was tried in several projects with varying degrees of success. Some experiences were

- People with a programming background could learn Adel in roughly two hours.
- Adel could be used not only for software design but also for logic design of hardware and operator parts of systems.
- The informal management of interfaces between objects (packages) made it necessary to perform manual checks on system description consistency.
- End users could learn to understand Adel descriptions if the description was given line by line.
- In software engineering, Adel helped sort out problems before programming commenced.

### Development of Odel

Experiences with Adel were encouraging, which led to the vision of a sys-

tem design language based on Ada 95. Although the new language was to provide system descriptions that could be used for programs written in any language, Ada was a logical foundation because it gives developers the ability to split a large system into packages, each of which contain a manageable and understandable part of the system with clear interfaces.

The idea was to create a language that was still simple enough to be taught to people with a programming background in a couple of hours and that was explainable to end users. At the same time, the language needed to be sufficiently formal that hardware engineers, software developers, and end users could check for consistency problems among the descriptions.

The resulting language, Odel, was developed by the private company Romet as part of the "complex systems" program sponsored by the Swedish Authority for Industrial Development. Romet retained ownership of Odel but decided to make the language publicly available without cost. Although Odel is derived from Ada 95, some constructs are different, and you cannot use Ada 95 tools to analyze Odel descriptions.

### Benefits of the Odel Language

Detailed system descriptions written in the Odel can be understood not only by users and developers but also by tools that support the Odel language. This

gives system designers the ability to “execute” the system description line by line—identifying many potential problems long before the system is implemented. Following are some of the tasks in which Odel can assist.

- Reveal incompleteness and inconsistencies in the understanding of a system that may be hidden in natural language design descriptions.
- Model a system so that you can estimate its feasibility.
- Use operator role descriptions as a base for writing operator manuals and for analyzing possible operator behavior, including erroneous behavior.
- Check a design for syntax and consistency errors and correct them.
- Find design contradictions.
- Find undefined states in a design.
- Investigate a system’s behavior under load without implementation of the system, which requires timing information for actions and capacity measurement for supporting hardware objects. This is best done with simulation first.
- Find out how system behavior depends on the operator’s actions within the relevant part of the operator’s behavior space.
- Create a basis to prepare tests for a system implementation, based on design understanding, possible inputs, and expected outputs.
- Create Fault Trees and Failure Mode Effects Trees for analysis of critical systems based on actions in Odel descriptions.
- Create a basis for definition of simulations.
- Compare software and hardware design descriptions with Ada 95 [1] VHDL (VHSIC Hardware Description Language [2]), and C++ (or Java) code. (This is a long-term objective for Odel, as it requires experience before implementation in a tool.)
- Transform Ada 95 source code into a design language description (long-term objective, probably requiring Ada Semantic Interface Specification [3]).

- Create a formal basis for building software tools for work with Odel descriptions.
- Teach Odel to software developers in a couple of hours.
- Explain Odel descriptions to end users, with full understanding. Descriptions of operator’s roles and hardware are special exceptions to the above, as they are not intended to be developed into code except for cases in which you want software to model hardware or operator behavior.

## Overview of Odel

### Overall Structure

Figures 1-3 show the overall structure of an Odel description. The central element in an Odel description is the *action*. The word “action” was chosen based on an idea from Professor Vitalis Sh Kaufman from Helsingfors, Finland, and it does not mean anything in the Ada programming language. The “action” represents Ada’s procedures, functions, and tasks.

Figure 1 shows how an action contains interfaces and behavior and how it is supported by type definitions and by definitions of parameters, local variables, and messages.

Administratively, actions are grouped into the following objects (Figure 2).

- **Ordinary objects** – contain a number of actions to be kept together for design and review.
- **Configuration Item object (CI object)** – encompasses the amount of work suited for a small group during a limited time.
- **Project object** – encompasses the complete work within a development project.

### Presuppositions for Odel

A number of presuppositions were formed as a basis for Odel:

- Any information system is constituted of one or more processes, which can be active in parallel (concurrently).
- Any process can be defined as an *action*, which has
  - An offered interface (action call).

- A behavior, defined as a sequence of statements, that may include sending and receiving messages.
- A required interface toward other actions (optional).
- Information system actions can be implemented as
  - operator actions.
  - software actions.
  - hardware actions.
- Each information system aims at fulfillment of one or more missions, with the relevant actions defined for each mission.
- Each mission is completed through one or more actions.

Figure 1. Overall Odel structure.

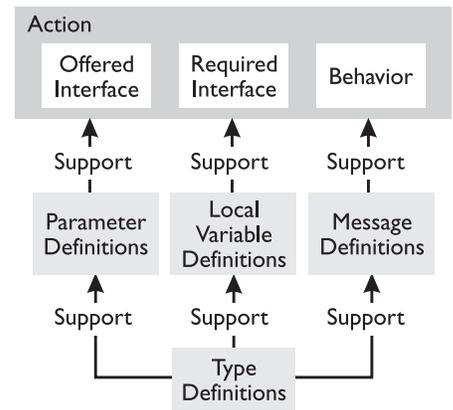


Figure 2. Action and object hierarchy.

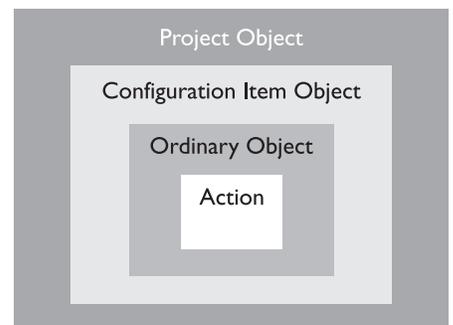
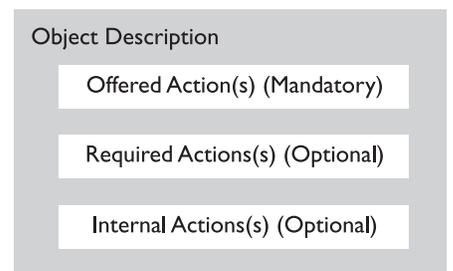


Figure 3. Object description.



- Ada 95 qualifies as a formal base for part of the definition of the Odel language.

## The Object with Its Interfaces

An ordinary Odel object is basically a container of actions (Figure 3):

**Offered action** – an action that can be invoked by an action outside the current object. “Outside” then includes other objects in the current CI and any object in its environment. At least one offered action is mandatory, as each object must have an offered interface. Parameters are defined in connection with actions.

**Required action(s)** – an action that the current object requires from objects outside the current object to complete its own actions. The required action(s) are optional, as “bottom-level” objects will not have any required interface. However, this entry is mandatory for all objects that have support objects. The required action names are qualified in action calls in two forms:

- *CI\_name.Object\_name.Action\_name (parameters)*, for actions defined in an object in a foreign CI. Objects in foreign CIs are called “attached objects.”
- *Object\_name.Action\_name (parameters)*, for actions defined in another object in the same CI as the current object. Objects within the current CI are called “contained objects.”

**Internal actions** – are only invoked from other actions within the current object.

The textual form for an object is  
 object *current\_object\_name* is  
 required actions : list of qualified names of actions  
 offered action : list of names of actions in current object with visibility = offered  
 internal actions : list of names of actions in current object with visibility = internal list of action descriptions  
 end *current\_object\_name*

## The Odel Definition

Because Odel is based on Ada 95, it was defined through copying and referencing the Ada 95 Language Reference Manual [1]. The main part of the language is merely a subset of Ada 95.

However, some extensions were needed to describe parallel processes. An example from the Odel definition is shown for the “Send” and “Receive” statements:

### Send Statements

A send statement assigns a value to a message and makes it globally readable.  
 send\_statement ::= send message\_name (expression)

The message denoted *message\_name* must be declared, and the expression must be of the type of the message.

Example:

In action header:

Messages: notification : string

In Odel description:

send notification (“OK”)

### Receive Statements

A receive statement retrieves a value from a message sent by another (or possibly the same) action and assigns it to a variable, an out parameter or an in\_out parameter.

receive\_statement ::= receive message\_name (variable | parameter)

The message denoted *message\_name* must be declared. The type of the variable or parameter must be the type of the message. For example:  
 receive notification (answer).

In real-time systems, you sometimes need to wait for a new message and then receive the new message. This can be expressed with a comment. Another, more formal possibility is to construct a loop and compare received messages until a new message is found. This possibility is, however, not normally recommended, as the loop may become obscure and unnecessarily prescribe a detailed programming solution.

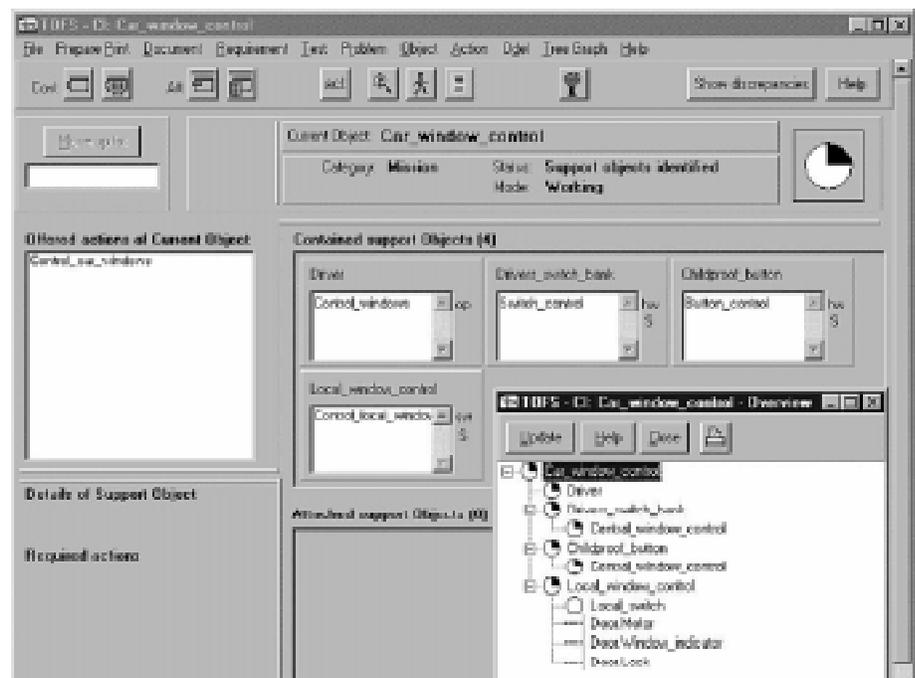
## Tofs, the Odel-Based Tool

The Odel language was used as a foundation to create a system engineering tool, Tofs (TOol For Systems). Figure 4 shows Tofs’ main screen with a “tree graph” window. This main screen allows you to work with the system structure and allows you to reach different parts of the Tofs toolkit.

To work with Odel, you select an action and open an editing window. After you have written some Odel, you can invoke the analyzer, which will then analyze the Odel description and then show any error messages by highlighting the lines where errors were found.

Figure 5 shows the Odel editing window, with a message from the analyzer. Tofs further includes an “execu-

Figure 4. Tofs main screen with automobile example.



tor” (not shown in the figures), which is basically an Odel-level debugger, which shows concurrent actions in multiple windows in parallel.

### Use of Odel and Tofs

With Odel and Tofs, the basic semantics and parts of the syntax of Ada 95 can be applied not only to software but also to complex concurrent systems where operators jointly operate with software and hardware modules to complete missions. Odel and Tofs are intended to support a system engineer in tasks such as

- Analyzing a system in its environment, with clarification of how the system communicates with and represents its environment.
- Designing a system as a set of objects that depend on each other through interfaces.
- Analyzing requirements and distributing fulfillment requirements to objects with listing of requirements with tracing.
- Managing problems that surface during system development.
- Analyzing a design concerning syntactic correctness and consistency.
- Reviewing a system, using the Odel “executor.”
- Documenting a system, according to relevant standards, such as the EIA/IEEE 12207.
- Reengineering code (manually) into an Odel system description.
- Analyzing a system’s dependability, using fault tree analysis and failure mode affects analysis.

Although Odel descriptions look similar to Ada code, they are not code but are a detailed and formal description of a system, including its software parts. To go from an Odel description to code, you can copy the Odel text into a source code editor and mark the text as comments. The Odel description will then be a detailed specification for the code to be written (in Ada or some other language).

Automatic generation of code has not been attempted because I believe that programming is professional work, which is best done by a compe-

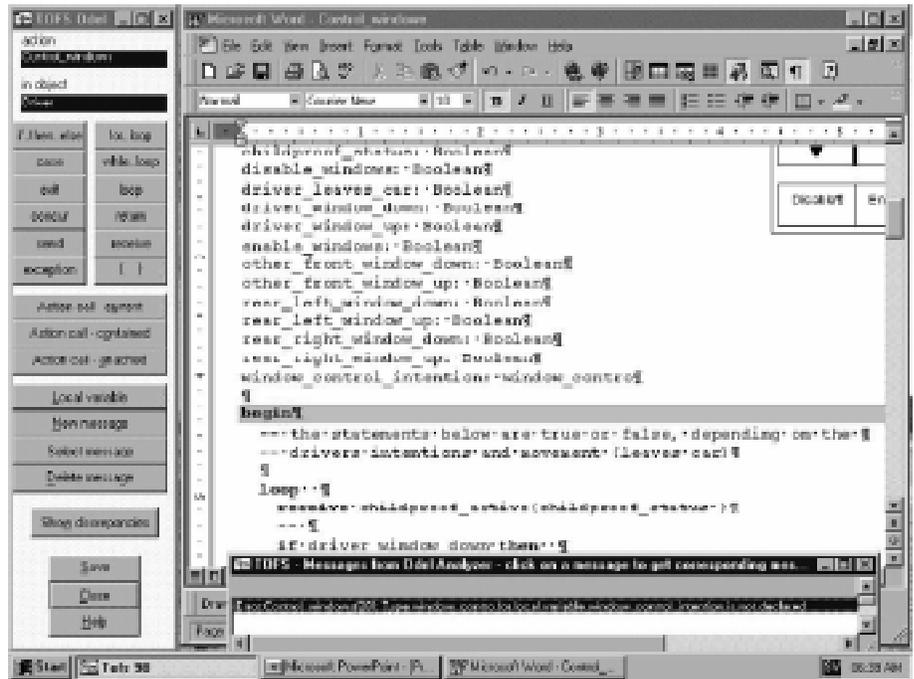


Figure 5. Odel editing window with error message.

tent programmer who understands what is required.

### Ownership, Availability, and Experience

The Tofs toolkit was developed by the Tofs AB company. The toolkit is available in two versions:

- A free version, with full functionality but limited capacity, intended for systems engineering education and for evaluation. It can be downloaded from <http://www.toolforsystems.com>
- A commercial version.

Courses in systems engineering with Tofs are held regularly in Sweden. Course length is three days. Some courses have also been held in the United States, and courses can be arranged on request by Abelia in Fairfax, Va. [4]. Some educational information also is available for download from <http://www.toolforsystems.com>

As the work with Ada-based systems engineering has been going on for more than 10 years, some experience has been gathered from various projects such as automobile industrial systems, military ship modeling (submarine), development of an Army radio communication system, structur-

ing of a tactical aircraft simulator, and others. Following are some experiences from these applications.

- Some end users readily accept and understand the formality of Odel without additional explanation from system developers, while the reaction of others is “I do not read code.”
- Using the Odel language without an automatic analyzer requires much tedious work for the reviewers. Tofs eliminates much of this type of work.
- Odel is useful when you want to document existing software, for example, for reengineering from FORTRAN into Ada.
- Inclusion of “mission objects” in the system structure helps to keep the priorities right and to understand how different parts of the system contribute to completion of missions.
- Inclusion of operator roles as objects in the system structure helps in optimizing human-machine interfaces.

### Conclusion

Odel has created the ability to structure complex systems as a set of ob-

jects, connected through their offered and required interfaces.

Experience from multiple projects shows that work with complex systems should be assisted by software tools, because the complexity results in more information than can be managed manually. Some 10 years of experience of Ada-based systems engineering has been used to implement the Tofs toolkit. The result is a comparatively compact tool to assist systems engineering tasks from requirements management to documentation.

The main objective for Odel was to create a language that qualifies as formal while being understood by both end users and developers. This objective seems to have been reached, but

more experience is needed to attain the full potential of the language. ♦

#### About the Author

**Ingmar Ögren** is president of Tofs AB in Sweden. His systems design experience began at the Royal University of Technology, Electronics Division, where he participated in pioneering military aircraft simulator systems. He later was active in the development of data transmission elements for air defense systems and the management of a multi-industry, multiprocessor airborne computer system. As a private company owner, he worked on civilian systems and helped introduce Ada to the Swedish defense community and helped use Ada to analyze, design, and document systems in general. This work led to his

development of the O4S™ method used in large systems such as submarines and aircraft. He and his wife Anna used their O4S experience as a foundation for Tofs.

E-mail: [iog@toolforsystems.com](mailto:iog@toolforsystems.com)  
Internet: <http://www.toolforsystems.com>

#### References

1. Ada 95 Reference Manual, ISO/IEC 8652, 1995.
2. IEEE Standard VHDL Language Reference Manual, ANSI/IEEE STD 1076-1993.
3. Information about ASIS is available from the URL <http://www.ci.pwr.wroc.pl/cgi-bin/plcon/winHiso/www.acm.org/sigada/wg/asiswg>.
4. Abelia is at <http://www.abelia.com>.

## Mark Your Calendars Now for the Eleventh Annual Software Technology Conference

May 2–6, 1999, Salt Lake City, Utah

It's not too early to start making plans to attend STC '99. Judging from past conference attendance, the sooner you act the better. Contact us for the latest information on conference registration, exhibits, housing, and activities in the host city. If you have never attended—or if you want to know how this year's event will be different—we'll give you an update on speakers, networking opportunities, general sessions, tutorials, presentation tracks, and exhibitors.

The United States Air Force, Army, Navy, Marine Corps, and the Defense Information Systems Agency have again joined forces to

co-sponsor STC '99, the premier Software Technology Conference in the Department of Defense. Utah State University Extension is the conference non-federal co-sponsor. We anticipate over 3,500 participants from the services, other government agencies, contractors, industry, and academia.

Registration for exhibit space opened Aug. 3, 1998. Reservations for exhibit space are processed on a first-come, first-served basis, by either mail or fax. Many exhibitors have already made reservations, and booth space is filling fast. New exhibitors will find background information on the Software Technology Conference, its history, and attendance statistics helpful in planning for the conference. This information, along with an updated exhibit hall layout, including assignments and organizations registered to date, will be maintained on the Internet. You can access this information at <http://www.stc-online.org> or from our STC '99 fax-on-demand line, 435-797-2358, item 303.

It is recommended that you make reservations for your hotel guest room as soon as possible. For your convenience, a Housing Reservation Form has been included in the center insert of this issue of *CROSSTALK*. In addition to the housing form, a listing of the potential hotels is available at our Web site and from our fax-on-demand line, 435-797-2358, item 200. Although this year more government-rate rooms have been made available in the immediate vicinity of the Convention Center, they will fill up quickly, so book early.

For more information, please visit our Web site at <http://www.stc-online.org>, call the management team at 801-777-7411, DSN 777-7411, or send an E-mail to Dana Dovenbarger or Lynne Wade at [wadel@software.hill.af.mil](mailto:wadel@software.hill.af.mil). We look forward to seeing you next May.





## Taking 3.1 Steps Forward, 98 Steps Back

The above title illustrates a point, but it is not to dump on Microsoft. A journal for technology-savvy readers is no place to make glib, rash statements about complicated technology issues. Instead, this column shows how a conspiracy by the personal computer (PC) industry is turning us into profitable zombie slaves.

I see no other reasonable explanation for why the world's bean counters, who usually hold their organizations' purse strings tightly enough to asphyxiate a musk ox—who create databases that track trends in staple usage—can blow thousands of dollars per employee per year on new computer stuff, and they don't seem to care whether they're getting a return on their investment.

Well, I've performed a study on our returns that should zap them back to reality. Of course, my startling conclusions would be meaningless without good data, but as my co-workers will testify, I am full of it. My study, which contrasts my recent computer upgrade with my recollections of previous performance—and includes figures I computed on a *real calculator*—shows how my recent upgrade robbed me of four years productivity *in the first two days alone*.

Here's the math: My old computer had plenty of RAM and storage space, and at 200 MHz, executed most commands instantaneously. This makes sense, considering that, like many people, I only occasionally need to create real-time 3-D galactic radiation entropy data models. For writing documents, creating simple graphics, and doing E-mail, 200 million cycles per second was kind of overkill.

So, seeing that my machine was more than adequate, and was set up with everything I needed, my organization decided to do something about it. They gave me a 300 MHz machine that has the sole advantage of providing *50 percent more overkill*. Well, maybe that's unfair. My larger documents are now printed a full 60 seconds before I mosey over to the printer—not just 55 seconds earlier. Also, I'm saving a couple of seconds each time I re-boot the system, which greatly helps, since the new computer crashes several times each day.

All told, the new computer may save up to 40 seconds per workday over the old one. However, it took me 12 hours to get it configured the way I need it, which means I'll have to work on it for four and a half years to get my 12 hours back. But that's a moot point, because next year I'm getting an *even more* instantaneous computer and a new operating system to learn. I'll soon be hundreds of years behind, but at least they'll be paying thousands of dollars for the privilege.

Perhaps you think the practice of buying marginally useful hardware and software upgrades is partially due to clever industry marketing and licensing practices. However, I have discovered that there is a much more reasonable explanation for our bizarre purchasing behavior: brainwave-altering voodoo satellites.

Here's how the satellites work: Say company X creates a word-processing upgrade with two exciting new features: automatic paragraph formatting you don't want but don't know how to undo, plus a new format no other application can read. The company then uses these satellites to broadcast a powerful voodoo spell: "There's a higher version number available for product X, you technology fossils! You must upgrade now!" A tiny, powerful zombie army then forms a line at the software store, eager to start distributing files nobody can open.

The rest of us receive these files, get sick of asking people to resend a readable version, and decide it's easier to upgrade our office suites, since there's a budget for upgrades anyway. We then in turn share our files with others who in turn have to get upgrades. A few months and perhaps billions of dollars later, everyone has upgraded entire software suites in order to gain the ability to open one type of file. Few people seem to be bothered by this, despite the fact that nobody on the planet has intentionally used a new word-processing feature since 1992.

Therefore, my column's title illustrates that the newest, latest technology isn't necessarily a step forward in terms of productivity. Imagine our joy, when the next upgrade wave came along, if everyone just said, "Nah, maybe next time." I doubt we'd miss out on much. Further details are available in my full study, which is available in a 3-D, full-motion interactive multimedia presentation (DVD-format only). You'll probably need an upgrade to view it.

—Lorin May

*Got an idea for BACKTALK? Send an E-mail to [backtalk@stsc1.hill.af.mil](mailto:backtalk@stsc1.hill.af.mil).*

<b>Sponsor</b>	<b>Lt. Col. Joe Jarzombek</b> 801-777-2435 DSN 777-2435 <a href="mailto:jarzombj@software.hill.af.mil">jarzombj@software.hill.af.mil</a>
<b>Publisher</b>	<b>Reuel S. Alder</b> 801-777-2550 DSN 777-2550 <a href="mailto:publisher@stsc1.hill.af.mil">publisher@stsc1.hill.af.mil</a>
<b>Managing Editor</b>	<b>Forrest Brown</b> 801-777-9239 DSN 777-9239 <a href="mailto:managing_editor@stsc1.hill.af.mil">managing_editor@stsc1.hill.af.mil</a>
<b>Senior Editor</b>	<b>Sandi Gaskin</b> 801-777-9722 DSN 777-9722 <a href="mailto:senior_editor@stsc1.hill.af.mil">senior_editor@stsc1.hill.af.mil</a>
<b>Graphics and Design</b>	<b>Kent Hepworth</b> 801-775-5798 <a href="mailto:graphics@stsc1.hill.af.mil">graphics@stsc1.hill.af.mil</a>
<b>Associate Editor</b>	<b>Lorin J. May</b> 801-775-5799 <a href="mailto:backtalk@stsc1.hill.af.mil">backtalk@stsc1.hill.af.mil</a>
<b>Editorial Assistant</b>	<b>Bonnie May</b> 801-777-8045 <a href="mailto:mayb@software.hill.af.mil">mayb@software.hill.af.mil</a>
<b>Features Coordinator</b>	<b>Denise Sagel</b> 801-775-4396 <a href="mailto:features@stsc1.hill.af.mil">features@stsc1.hill.af.mil</a>
<b>Customer Service</b>	801-775-4396 <a href="mailto:custserv@software.hill.af.mil">custserv@software.hill.af.mil</a>
<b>Fax</b>	801-777-8069 DSN 777-8069
<b>STSC On-Line</b>	<a href="http://www.stsc.hill.af.mil">http://www.stsc.hill.af.mil</a>
<b>CROSSTALK On-Line</b>	<a href="http://www.stsc.hill.af.mil/Crosstalk/crosstalk.html">http://www.stsc.hill.af.mil/Crosstalk/crosstalk.html</a>
<b>ESIP On-Line</b>	<a href="http://www.esip.hill.af.mil">http://www.esip.hill.af.mil</a>

**Subscriptions:** Send correspondence concerning subscriptions and changes of address to the following address:

Ogden ALC/TISE  
7278 Fourth Street  
Hill AFB, UT 84056-5205

E-mail: [custserv@software.hill.af.mil](mailto:custserv@software.hill.af.mil)  
Voice: 801-775-4396  
Fax: 801-777-8069 DSN 777-8069

**Editorial Matters:** Correspondence concerning *Letters to the Editor* or other editorial matters should be sent to the same address listed above to the attention of *CROSSTALK* Editor or send directly to the senior editor via the E-mail address also listed above.

**Article Submissions:** We welcome articles of interest to the defense software community. Articles must be approved by the *CROSSTALK* editorial board prior to publication. Please follow the *Guidelines for CROSSTALK Authors*, available upon request. We do not pay for submissions. Articles published in *CROSSTALK* remain the property of the authors and may be submitted to other publications.

**Reprints and Permissions:** Requests for reprints must be requested from the author or the copyright holder. Please coordinate your request with *CROSSTALK*.

**Trademarks and Endorsements:** All product names referenced in this issue are trademarks of their companies. The mention of a product or business in *CROSSTALK* does not constitute an endorsement by the Software Technology Support Center (STSC), the Department of Defense, or any other government agency. The opinions expressed represent the viewpoints of the authors and are not necessarily those of the Department of Defense.

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

**STSC On-Line Services:** STSC On-Line Services can be reached on the Internet. World Wide Web access is at <http://www.stsc.hill.af.mil>. The STSC maintains a Gopher server at [gopher://gopher.stsc.hill.af.mil](http://gopher.stsc.hill.af.mil). Its ftp site may be reached at <ftp://ftp.stsc.hill.af.mil>. The Lynx browser or gopher server can also be reached using telnet at [bbs.stsc.hill.af.mil](telnet://bbs.stsc.hill.af.mil) or by modem at 801-774-6509 or DSN 775-3602. Call 801-777-7989 or DSN 777-7989 for assistance, or E-mail to [schreifr@software.hill.af.mil](mailto:schreifr@software.hill.af.mil).

**Publications Available:** The STSC provides various publications at no charge to the defense software community. Fill out the Request for STSC Services card in the center of this issue and mail or fax it to us. If the card is missing, call Customer Service at the numbers shown above, and we will send you a form or take your request by phone. The STSC sometimes has extra paper copies of back issues of *CROSSTALK* free of charge. If you would like a copy of the printed edition of this or another issue of *CROSSTALK*, or would like to subscribe, please contact the customer service address listed above.

The **Software Technology Support Center** was established at Ogden Air Logistics Center (AFMC) by Headquarters U.S. Air Force to help Air Force software organizations identify, evaluate, and adopt technologies that will improve the quality of their software products, their efficiency in producing them, and their ability to accurately predict the cost and schedule of their delivery. *CROSSTALK* is assembled, printed, and distributed by the Defense Automated Printing Service, Hill AFB, UT 84056. *CROSSTALK* is distributed without charge to individuals actively involved in the defense software development process.