How long can you wait for CMMI® Compliance?

Manage your software and systems engineering projects in guaranteed compliance with the CMMI NOW!

processMax 2i and 3i include all the necessary policies, procedures, guidelines, criteria, templates, and forms in role-based, step-by-step instructions, ready for use—everything you need for compliance with the CMMI-DEV at Level 2 and Level 3 respectively. Integrated with robust document management, workflow, and automated measurement and reporting, processMax is the intranet web-based solution for effective and efficient management of your software and systems engineering projects. processMax enables you to start operating in compliance with the CMMI immediately! We guarantee that organizations using processMax will not fail a formal appraisal led by an independent SEI-licensed appraiser.

pragma SYSTEMS CORPORATION
www.pragmasystems.com
877.838.PMAX
info@pragmasystems.com

GSA Schedule Contract No. GS-35F-0559S. processMax is a registered trademark of pragma SYSTEMS CORPORATION. Although processMax makes use of portions of "CMMI for Development, Version 1.2," CMU/SEI-2006-TR-008, copyright 2006 by Carnegie Mellon University, neither the Software Engineering Institute nor Carnegie Mellon University have reviewed or endorsed this product.

Copyright © 2008 pragma SYSTEMS CORPORATION.
The focus of this issue of Software Tech News is technology challenges related to software engineering and technology within the DoD. The DoD is faced with enormous challenges in developing and acquiring software intensive systems. This issue examines the challenges, needs, and some of the research activities that are ongoing to address the producibility of software.

Coincidentally, the Defense Technical Information Center (DTIC) recently launched a wiki-based online collaborative encyclopedia for scientists, engineers, program managers and warfighters called DoDTechpedia (https://www.DoDTechpedia.mil). DoDTechpedia (see Figure 1) is a live forum for users to explore and discuss innovative and emerging technologies from the academic and private sectors, as well as to discuss technical solutions to DoD challenges.

The richest area of information on the DoDTechpedia appears by clicking on the Technology Areas link in the Navigation box. Clicking on that link exposes the reader to more than 40 technical areas of interest to the DoD. Two technology areas, Software Development Technology and Information Technologies are of particular interest to readers of the DACS STN.

The Software Development Technology entry (Figure 2) is of interest because it highlights the growing software capabilities needed by the DoD to meet ultra-large system demands; software interchangeability and interoperability requirements; use of COTS, open systems and standards to facilitate interoperability; and new technology insertion needs to make future weapons systems more affordable. Of relevance to this issue of the STN, this DoDTechpedia entry also points out that the DoD has recognized that it is behind the curve on software and identifies a number of R&D software initiatives that have been undertaken by the DoD that are coincidentally discussed in articles in this issue of Software Tech News:

- A Software-Intensive Systems Productivity Initiative (SISPI) roadmap has been developed to foster technology research and transition to improve the acquisition, development, and advancement of software-intensive systems. Mr. Grady Campbell of the SEI, one of the lead authors of this roadmap, discusses the roadmap in his article “Advancing Producibility for Software-intensive Systems”.
- The Software Engineering Institute has recently published the results of a study addressing future needs of what are being called Ultra-Large Systems (ULS). As the report says, “These systems will push far beyond the size of today’s systems and systems of systems by every measure.” Ms. Linda Northrop of the SEI was a key author and lead investigator of this study and reports on the ULS study and the resulting research agenda proposed in her article “Ultra-Large-Scale Systems: Challenges and Promising Research Areas.”

Mr. Robert Gold, Director for Engineering, Prototyping and Technology Integration with DUSD (S&T), is a sponsor of the SISPI initiative discussed on this DoDTechpedia page. In this STN, Mr. Gold provides our lead article with a discussion of “Software Technologies in Technology Readiness Assessments for DoD Acquisition Systems” in which he discusses how
software technology manifests itself in major defense acquisition systems.

In “The Graduate Software Engineering Reference Curriculum: A Joint Industry, Government and Academic Project,” Dr. Richard Turner and Dr. Art Pyster, Stevens Institute of Technology, provide us a summary of a DoD funded initiative to define entry and exit criteria for a Master’s Degree program in Software Engineering.

The Information Technologies technology area (Figure 3) on DoDTechpedia is a closely related area because it discusses federal initiatives in key software intensive technical areas such as net-centric applications; information fusion, management, and distribution; wireless network security; computer network operations; information assurance; high productivity computing systems; and modeling and simulation. Many initiatives are discussed on DoDTechpedia, but of particular relevance to this issue of STN is a discussion of the Networking and Information Technology Research and Development (NITRD) Program. One of the references discusses the 2008 R&D budget plans of the 14 member NITRD Federal agencies that support R&D in advanced computing, networking, cyber security, software, and other information technologies.

I would encourage the readers of STN to browse in-depth the information presented on DoDTechpedia if the reader is interested in understanding and collaborating with the DoD on software and IT research initiatives.

On a separate subject, we have recently received permission to publish an article that was originally scheduled to be published in our March 2007 STN issue on “Performance Results from Process Improvement”. This article, titled “Performance Outcomes of CMMI-Based Process Improvement at SAIC Defense Solutions Group” and co-authored by Sharon Cobb Flanagan of SAIC and Dennis Goldenson of the SEI, discusses SAIC’s performance results in applying CMMI. We do not normally mix topics in the STN, but we felt it important that readers be able to read these new findings.

About the Author

Thomas McGibbon works for ITT Advanced Engineering & Sciences as the Director of the Data & Analysis Center for Software (DACS) since 1994. He has over 30 years of experience in software development, systems development, and software project management. He is author of several DACS state of the art reports on software engineering topics. He holds a MS in Software Engineering from Southern Methodist University and BS in Mathematics from Clarkson University. He is a Certified Software Development Professional (CSDP).

Author Contact Information
Email: tom.mcgibbon@itt.com
Let’s face it; software is difficult no matter how you slice it. Consideration of software technology maturity as part of technology readiness is even more difficult because it requires quantifying the underpinnings of the one part of our defense systems which, by its very nature, has few natural bounds and is almost infinitely malleable. Fortunately, we have built up a body of experience and knowledge as a result of several years of conducting Technology Readiness Assessments (TRA) for major defense acquisition programs of all sorts – embedded, infrastructure, net-reliant, and business systems. The corporate knowledge is by no means perfect but this is an excellent time to document what we know, share it as a community, and figure out what could be done better in the future. My experiences and insights are noted below; please add yours to the community through your Service and Agency representatives.

Based on the TRAs I’ve reviewed and supported, software manifests itself as a technology in five distinct ways: **Unprecedented Functionality, Off-The-Shelf Components, Enabling Run-Time, Aggregation of Components, and Enabling Development.** I’ll discuss each type briefly and provide examples in the remainder of the article.

**Unprecedented Functionality** – We usually go through the effort to establish and execute a major defense acquisition program to get something of functional significance we didn’t previously have. This means a large portion of the associated development will attempt to provide capabilities to the war fighter we haven’t previously built. Most times, currently and in the foreseeable future, that new functionality is provided or critically enabled by the software. What’s important here isn’t the software though; it’s the algorithms or methods assembled by domain systems experts captured in the software which should be considered a new or novel technology. Because that technology is embodied in the software, maturity of the technology can, perhaps incorrectly, become synonymous with maturity of the software. What’s critical in these instances is not whether or not the software is documented, is produced by a Capability Maturity Model Integration (CMMI) level 3 developer, or has been formally qualified. Those considerations, while important, can obfuscate the real technology issues. The relevant issue for a TRA maturity evaluation is – are the new algorithms, logic or methods really suitable to accomplish the desired mission in the intended environment? As an example, the Space Based Surveillance System (SBSS) is an upgrade to the Mid-course Space Experiment (MSX) program for observing and cataloging objects floating in near earth orbit. SBSS’s sensor supports significantly more tasking than its predecessor’s sensor such that manual tasking of the satellite would be the limiting factor in tracking objects, not the capability of the platform. To address this shortcoming, the Massachusetts Institute of Technology Lincoln Laboratory developed a set of automated sensor tasking algorithms allowing incoming tasks to be prioritized and scheduled by the ground system in a manner that optimized sensor movement. To mature these algorithms, Lincoln Laboratory conducted a series of lab demonstrations showing how emergent tasks could rapidly be handled efficiently and effectively.

**Off-The-Shelf (OTS) Components** – Some acquisition programs use already developed software components – government, commercial, open source – in a new or novel way. These OTS items are a cost-effective way for us to get the capabilities we need provided they are suitable for use in a military environment. Military run-time environments may differ from those envisioned by the product developer in the areas of information security, robustness, and compatibility with legacy applications. It’s typically establishing ability to perform in a relevant environment that becomes critical in assessing maturity of OTS technologies. OTS technologies are selected because there is already a body of experience or expectation the technologies will perform as intended. Will they be able to provide that capability alongside other technologies, or in conjunction with legacy applications? As with most technologies, hands-on experience in the actual or representative intended environment is necessary to establish technology maturity. The Army’s Global Command Support System (GCSS-A) is an SAP*-based enterprise management platform.
system intended, in part, to deploy with troops and battalions. To support the separation of a portion of the SAP database while units are deployed, the GCSS-A office elected to use the SAP Mobile Infrastructure (MI) bolt-on. This software feature hadn’t been used in previous instantiations of SAP in the DoD so we couldn’t assess its maturity based on previous use by the Department. To provide a basis upon which to conclude the component was mature, the Army prototyped the SAP MI bolt-on in a field demonstration.

Enabling Run Time – Not only do we attempt to build systems with new, unprecedented features, we sometimes organize existing capabilities in more efficient ways to gain other benefits such as reduced manning. These additional motivations for systems development sometimes necessitate the use of emerging infrastructure technologies to enable those improvements. The term ‘Enabling Run Time’ here means software technologies which are not directly visible to an operator or providing an end effect but are needed within the bowels of the system to enable what is done by the applications running on the system. These technologies might be middleware, features in an operating system, or a new approach for data management. As with any other critical technology, assessing suitability and characterizing capabilities and limitations are required to ensure the developers are fully informed as they design the system and build the software. An excellent example is the use of no-heap real-time (NHRT) Java in the DDG-1000 destroyer program. The DDG-1000 program elected to use Java in their Total Ship Computing Environment (TSCE) to gain the productivity benefits of the rich java libraries as well as enable increased isolation of software from hardware. Because many of the war fighting functions within the TSCE have hard or soft timing requirements, an ability to predict and manage deadlines within some of the Java implementations was required. No-heap real-time Java was selected as the default implementation for functions requiring real time performance. The system architects realized NHRT Java may not be capable of meeting all the hard real time performance requirements so they characterized the performance limitations of NHRT Java, assessed the timing needs of functions across the TSCE, and provided a small number (~3-4) of alternative implementations to allow subsystems developers to select an approach to meet their needs while maximizing the benefits of portability and OTS libraries throughout the system.

Aggregation of Components – In larger developments, it can be easier to manage software technology risks and expectations throughout oversight and program management by bundling technology issues into larger system or component oriented elements. (Note this same approach is often used in hardware technologies.) One of the best examples of how this might be employed in software is the above mentioned TSCE system for the DDG-1000. The TSCE is designed to allow the DDG-1000 to be operated with significantly reduced manning over previous ships. However, the Technology Readiness Assessment (TRA) for DDG-1000 didn’t include NHRT Java as an explicit Critical Technology Element (CTE). Instead, the TSCE itself was declared as the CTE commensurate with the way hardware technology challenges such as the Integrated Power System and the Dual Band Radar Suite were documented. The maturation of the TSCE as a CTE was measured through the successful delivery and certification of the software builds. Technology Readiness Level (TRL) 6 was declared with the certification of build 3 (of 6) because almost all of the significant technology challenges, including the use of NHRT Java noted above, were addressed and demonstrated by the functionality planned through that third build. The lower level technology challenges, of which NHRT Java was one, were acknowledged, documented, and discussed in regular status meetings between the DDG-1000 Program Office staff, software and systems leads from the developers, and the Office of the Secretary of Defense TRA oversight staff. The documented CTE analysis was done by assessing the TCSE.

Enabling Development – The final manifestation of I believe software, in technology discussions, is a little bit like metal. It has become such a fundamental building block of our complex warfighting systems that technology deliberations about software or metal as a generic element are meaningless.
software technologies I’ve experienced to date is software technologies which are critical to the development and evolution of a system but aren’t part of the deployed system. Technologies in this category might be used to enable better programming, management or sustainment of the software but do not have a material presence in the run-time deployment of the software and system. They may be also used in support of development activities such as modeling and simulation or verification and validation. Because these technologies are not deployed, they may not be found by examining the Work Breakdown Structures (WBS) or technical requirements for the system. One example of this type of software technology comes from the Acquisition Strategy (AcqStrat) of the Single Integrated Air Picture (SIAP) program. The SIAP AcqStrat noted that the incorporation of the SIAP software into multiple Air and Missile Defense systems would be facilitated through the use of the Model Driven Architecture® (MDA) methodology and toolset. Using MDA allowed the SIAP program office to centrally configuration manage algorithms and logic across the enterprise without the difficult burden of either providing implementations specific to all the potential run-time environments or requiring identical run-time environments across multiple major defense acquisition programs. To assess the maturity of MDA, the SIAP program office used this technology as an integral part of the experimentation phase of the program allowing the deployed system developers to gain experience with the tools and providing a basis upon which the enterprise could understand what capabilities this technology could really provide. MDA was declared a formal CTE within the SIAP TRA and ongoing assessments of the MDA technology and the toolsets are provided on a regular basis to oversight activities. During the technology assessment process, an assessment of the leading MDA toolsets resulted in a switch from one toolset to another to increase the robustness of the tools and allow the program to keep pace with ongoing technology improvements.

Conclusion - I believe software, in technology discussions, is a little bit like metal. It has become such a fundamental building block of our complex war fighting systems that technology deliberations about software or metal as a generic element are meaningless. We would certainly find it less than useful to have a paragraph about the deployment of metal in a war fighting system in a TRA. If a war fighting system had an issue with the properties of a particular alloy or the manufacturing of a specific type of metal, those concerns could be worthy of declaring it as CTE and funding the appropriate technology maturation activities. Likewise, understanding the specific needs or concerns about a way in which software is employed within a system or supports its development and deployment and conducting technology experimentation and maturation would be highly appropriate. It takes a confluence of software and systems expertise and vision to appreciate and articulate how software-related CTEs should be determined, documented and managed however.

These characterizations reflect what I’ve seen to date in TRAs but may not describe every circumstance encountered across the DoD. One of the emerging challenges not addressed above is the need to consider some technologies in a systems-of-systems or family-of-systems context. As we attempt to deploy technologies across multiple netted systems, assessing maturity in a relevant or operational environment becomes problematic. Accordingly, I highly encourage practitioners and participants in the TRA process to make their own observations and provide their insights through the Service and Agency TRA leads for consideration into future TRA Guidebook releases.

About the Author

Robert A. Gold is currently the Director for Engineering, Prototyping and Technology Integration, Deputy Under Secretary of Defense for Science and Technology (DUSD (S&T)).

He has over twenty two years of DoD acquisition experience and has focused on complex software-intensive system development and systems-of-systems integration for the last sixteen years. Mr. Gold began employment with Naval Sea Systems Command (NAVSEA) in 1986, where he served in a variety of systems engineering, software engineering, and acquisition positions for submarine, surface ship, and missile programs. Mr. Gold has served the Office of the Secretary of Defense since 2001 providing oversight of S&T, technical support to acquisition programs, and policy and guidance for technology readiness evaluations.

Mr. Gold is a member of the Professional Acquisition Workforce and is DAWIA level three certified in Systems Engineering, Science and Technology Management, and Program Management.

He holds a BSEE from Lehigh University, an MS in Systems Engineering from Virginia Polytechnic Institute, and a Public Management Certificate from Indiana University – Purdue University Indianapolis.

Author Contact Information

Email: Robert.Gold@osd.mil
Software quality and reliability are mission critical. The size, pervasiveness, and complexity of today’s software can push your delivery dates and budgets to the edge. That means rigorous quality discipline, and collaboration between development and QA teams has never been more important.

With offices worldwide and over 8,500 customers, Seapine Software is the leading provider of integrated quality-centric application lifecycle management (ALM) solutions. For over 13 years Seapine’s easy-to-use, award-winning tools have helped companies achieve success by streamlining communications, improving traceability, and making development and QA teams more productive.

Satisfy Your Quality Obsession.
www.seapine.com/gsa

GSA Advantage® GSA Schedule 70 Contract GS-35F-0168U
Integrated ALM for a Quality-Ready Advantage
Ultra-Large-Scale Systems: Challenges and Promising Research Areas

WE WILL FACE FUNDAMENTAL CHALLENGES IN THE DESIGN AND EVOLUTION, ORCHESTRATION AND CONTROL, AND MONITORING AND ASSESSMENT OF ULS SYSTEMS.

by Linda M. Northrop, Software Engineering Institute

Society’s dependence on software in every aspect of life is indisputable. From the Internet itself to Yahoo, Google, Wikipedia, and FaceBook, there is an unstoppable trend toward increasing scale in many of the systems that have become part of the way we conduct business, communicate, and fight wars. When coupled with the advances in wireless technology and more recently sentient technologies—chips that can sense, receive, store and transmit information—systems of unprecedented scale, ultra-large-scale systems, are emerging. They are emerging in areas that address many of society’s most vexing and pervasive issues: health care infrastructure, homeland security, transportation, power systems, energy conservation and climate tracking, and national defense. Nowhere is this more evident than in U.S. defense systems.

The U.S. Department of Defense (DoD) has a goal of information dominance—to achieve and exploit superior collection, fusion, analysis, and use of information to meet mission objectives. This goal depends on increasingly complex systems characterized by thousands of platforms, sensors, decision nodes, weapons, and war fighters connected through heterogeneous wired and wireless networks. These systems will push far beyond the size of today’s systems and systems of systems by every measure. They will be ultra-large, network-centric, real-time, cyber-physical, social systems that support a landscape of information-sharing constraints and must be adaptable and robust in extreme environments, and be sustainable legally, technically, and politically.

Given this trend, the office of the Assistant Secretary of the U.S. Army (Acquisition, Logistics, & Technology) (ASA ALT) funded the Software Engineering Institute (SEI) to lead a 12-month investigation of ultra-large-scale (ULS) systems. ASA ALT originally posed this question to the SEI: “Given the issues with today’s software engineering, how can we build the systems of the future that are likely to have billions of lines of code?” The study brought together experts in software and other disciplines and resulted in a report, Ultra-Large-Scale Systems: The Software Challenge of the Future [Northrop 06]. The report describes the characteristics and the challenges associated with ULS systems and then details a broad, multi-disciplinary research agenda for developing the ULS systems of the future.

What are ULS Systems?

The primary characteristic of ULS systems is ultra-large size on any imaginable dimension—number of lines of code; amount of data stored, accessed, manipulated and refined; number of connections and interdependencies; number of hardware elements; number of computational elements; number of system purposes and user perception of these purposes; number of routine processes, interactions, and emergent behaviors; number of (potentially overlapping) policy domains and enforceable mechanisms; and the number of people involved in some way—as consumers, producers, acquirers, or as actual elements in the system. ULS systems will be interdependent webs of software-intensive systems, people, policies, cultures, and economics.

But to understand the real nature of ULS systems, we must go beyond just the size. We must understand the effects of scale and the demands that ULS systems are likely to place on technologies, processes, and people. The prominent characteristics of ULS systems that arise because of their scale include:

- Decentralization: The scale of ULS systems will allow only limited possibilities for centralized or hierarchical control of data, development, evolution, and operation. Even the limited amount of hierarchical control that is possible today for very large systems and systems of systems will be challenged at the scale of ULS systems.

- Inherently conflicting, unknowable, and diverse requirements: The scale and complexity of problems to be solved by ULS systems mean that the requirements to be satisfied by the systems will not be adequately known until the systems are in use. Even then, as the system is put into operation, perceptions of the problems it is solving will change. Some problems addressed by ULS systems are likely to be so complex that requirements will never really converge.

- Continuous evolution and deployment: ULS systems will be in service for a long time. Their size will make it impractical to replace or retire them, and so they will continuously evolve to meet new and modified...
requirements and to incorporate new technologies. The evolution will need to occur while the system is operating. It will need to be accomplished by different groups of stakeholders, guided and constrained by rules and policies that allow local needs to be satisfied in local ways without destroying the integrity and value of the overall system.

- Heterogeneous, inconsistent, and changing elements: The sheer size of ULS systems will mean that its elements (hardware, software, procedures, rules, and people) will be heterogeneous, inconsistent, and changing. The elements will come from a variety of sources, some new and some old. Some may be dynamically supplied and assimilated. Many will be services that will be provided over the Internet. Because there will be so many sources for the constituent elements of the ULS system, there will be inconsistencies among those elements such as different computer languages; different versions of the same technology; and different methods, policies, and processes. And to complicate this further, the parts of the system will always be changing. New sensors will be incorporated, more software will be added, failed hardware will be replaced, and improved technology will be inserted. The exact composition of a ULS system cannot be known when it is being designed and may vary from moment to moment.

- Erosion of the people/systems boundary: There will be no distinct boundary between users, developers, and acquirers. Many whom we typically call users of today's systems will be involved in the actual development, extension, and adaptation of the system. Moreover, people will be part of the overall system behavior. For example, a soldier will become a distributed platform in the defense systems of the future.

- Normal failures: The vastness of physical underpinnings, the frequency of use, and the interconnections among software constituents that were not designed to work together will guarantee that failures will no longer be an exception. ULS systems will need to be designed to cope with failures as a continuous problem—to warn, to cordon off the effect of any failure, and to persist in the event of failure.

- New paradigms for acquisition and policy: Those responsible for making a ULS system possible (managers, acquirers, developers, suppliers, legislators, etc.) will be unable to comprehensively define and control the system using today's acquisition and policy approaches, which are predicated on centralized control.

These characteristics may appear in today's large systems and systems of systems, but in the ULS systems of the future they will dominate. More importantly, these characteristics undermine the key assumptions that underlie today's software engineering approaches—engineering approaches that work well for large-scale systems and will continue to work well for those same sorts of systems in the future. However, these top-down, plan-driven approaches will not work for ULS systems. They are based on the requirements/design/build cycle with standard well-defined processes, centrally controlled implementation and deployment, inherent validation and verification, and defect elimination, none of which scale to the new characteristics listed above that will be pervasive in ULS systems. Issues that are not significant at smaller scales become significant at ultra-large scales. These systems will surpass the thresholds at which today's approaches will work even nominally. Today's latest innovations, such as service-oriented approaches, agile development, and even open-source development may individually address some of the needs of ULS systems, but inadequately.

**Challenges of ULS Systems**

Fundamental gaps in our current understanding of software and software development at the scale of ULS systems present profound impediments to their technically and economically effective achievement. Hence there are profound impediments to realizing the DoD goal of deterrence and dominance based on information superiority. These gaps are strategic, not tactical. They are unlikely to be addressed adequately by incremental research within currently established categories. Rather, we require a broad new conception of both the nature of such systems and new ideas for how to develop them. We will need to look at them differently, not just as systems or systems of systems, but as socio-technical ecosystems. We will face fundamental challenges in the design and evolution, orchestration and control, and monitoring and assessment of ULS systems. Because the characteristics of ULS systems fundamentally undermine the assumptions of today's approaches, these challenges require breakthrough research.

**Research Agenda**

The study proposes a ULS systems research agenda for an interdisciplinary portfolio of research in at least the following areas:

- Human Interaction
- Computational Emergence
Design
Computational Engineering
Adaptive System Infrastructure
Adaptable and Predictable System Quality
Policy, Acquisition, and Management

The proposed research does not supplant current, important software and system research but rather significantly expands its scope. Because we are focused on systems of the future, we have purposely avoided couching our descriptions in terms of today's technology. The envisioned outcome of the proposed research is a spectrum of technologies and methods for developing these systems of the future, with national-security, economic, and societal benefits that extend far beyond ULS systems themselves.

Prioritizing and Carrying Out the Research

Though the research agenda proposed in the report does not prescribe a single, definitive roadmap for prioritizing and conducting the research, it does present three structures that suggest ways to cluster and prioritize research areas mapping the research areas and topics to

1) Specific DoD missions and required capabilities
2) DoD research funding types required to support them, and
3) Estimates of the relative starting points of the research.

These structures can then be used to define one or more roadmaps that could lead to one or more ULS-systems research programs or projects.

Recent Activity

Since the publication of the report in May 2006, there has been a growing community of interest and research starts focused on ULS systems. More than 100,000 copies of the report have been downloaded from the SEI website in addition to more than 3,800 paper copies distributed. Members of the study author team have given more than 25 keynote addresses and participated in press and industry-analyst reviews. There was a feature article on ULS systems in IEEE Software [Goth 2008].

A National Science Foundation Center on Ultra-Large-Scale Software-Intensive Systems has been established; research at participant universities (Vanderbilt University, University of Virginia, Michigan State University, University of California at San Diego) has been initiated in digital evolution, design for ULS systems, and adaptive system infrastructure. In the United Kingdom, the government has pledged the equivalent of approximately 20 million U.S. dollars to the Large Scale Complex IT Systems Initiative. There was a conference on ULS systems sponsored by the Strengthening the Mid-Atlantic Region for Tomorrow (SMART) organization and attended by leaders from DoD, DoE, DHS, and industry. There have been research workshops dedicated to ULS systems as part of three conferences (OOPSLA 2006, ICSE 2007, and ISCE 2008) that have involved more than 75 researchers from around the world who are doing research inspired by the ULS System Report. The SEI created a ULS System Research Roadmap, sponsored by Army CERDEC, which provides greater fidelity to the research agenda published in the report for the following capability: “Maintain coherent common operating picture by rapidly collecting, processing, disseminating, and protecting information spanning echelons, services, and coalitions across a mix of ultra-large-scale environments.” The SEI maintains a website that describes these and other activities focused on ULS systems [SEI 2008].

SEI Research

The SEI has initiated research in four areas identified in the report on ULS systems: the role and meaning of architecture in a ULS system context, distributive governance and acquisition in dynamic contexts, computational mechanism design, and most recently human interaction. The most significant results have been obtained in the area of computational mechanism design.

A mechanism is an institution such as an auction, voting protocol, or a market that defines the rules for how humans are allowed to interact and governs the procedure for how collective decisions are made. Economic mechanisms have a long history of being used for decentralized decision making by rational agents. Computational mechanisms arise where computational agents work on behalf of humans. A team of researchers at the SEI has investigated the potential for using computational mechanisms to improve the quality of a combat group's common operating picture, in a setting where network bandwidth is scarce. They developed a robust emulation of a tactical data network (based loosely on the Navy LINK-11) and developed a variant of a well-known auction mechanism to allocate network bandwidth for radar sensor fusion [Klein 2008]. They also conducted an experiment to allocate bandwidth in an ad hoc technical network in conjunction with the Center for Network Innovation and Experimentation (CENETIX) Technical Network Topology Experiments at Camp Roberts in California with the Special Operations Command (USSOCOM). This work demonstrates that economic mechanisms are a feasible and interesting
alternative to traditional systems approaches to resource allocation in systems that are highly dynamic. Centralized decision and control are mismatched with today’s large-scale, distributed systems. The principles of economics are applicable to the kinds of decentralized decision making required by network-centric systems. Computational mechanisms bring economic theory to the realm of software engineering to address human incentives in decentralized decision making. Though representing only one research topic in the many that were identified in the ULS system study, this research in computational mechanism design demonstrates the power of an interdisciplinary approach to large and ultra-large-scale systems.

Post-Study Observations

The trend toward ULS systems and the impact of scale are evident. We are technically not ready to deal with the impact of scale. It is important to note, however, that the study does not contend that all systems of the future will be ULS systems. Clearly, they won’t be. However, the systems key to sustaining ongoing transformations in national defense, to achieving information dominance, and to dealing with many of society’s critical issues will be.

Some believe that there are ULS systems today. Some argue that ULS systems are really systems of systems or complex net-centric systems. ULS systems certainly have characteristics in common with some systems of systems, especially those at the level Maier calls “virtual system of systems” [Maier 1996], but not all. There are some system-of-system engineering approaches that work for directed or acknowledged systems of systems that will not work for what we have described as ULS systems. In truth, what you call a system (system of system, ULS system, complex net-centric) is really unimportant. What is important is that ULS system characteristics are recognized when present because these characteristics undermine the assumptions we make in most current technical, management, and acquisition approaches. This ULS system perspective is helpful in understanding some of the current technology and management shortcomings and issues with systems of systems. We have also found through our research in computational mechanism design that some of the research identified in the ULS system study can also have a positive impact on systems that are not ultra large scale.

References


About the Author

Linda Northrop has more than 35 years of experience in software development as a practitioner, researcher, manager, consultant, and educator. She currently is director of the Research, Technology, and Systems Solution Program at the SEI where she leads the work in architecture-centric engineering, software product lines, performance critical systems, systems of systems, and ultra-large-scale (ULS) systems. She is coauthor of the book Software Product Lines: Practices and Patterns and led the research group on ULS systems that resulted in the book, Ultra-Large-Scale Systems: The Software Challenge of the Future.

Before joining the SEI, she was associated with both the United States Air Force Academy and the State University of New York as professor of computer science, and with both Eastman Kodak and IBM as a software engineer. She is a recipient of the Carnegie Science Award of Excellence for Information Technology, the New York State Chancellor’s Award for Excellence in Teaching, and the ACM SIGPLAN Distinguished Service Award.

Author Contact Information

Email: lmn@sei.cmu.edu
Advancing Producibility for Software-Intensive Systems

THE PRIMARY MOTIVATION FOR SISPI IS TO IDENTIFY AND SOLVE THE UNDERLYING PROBLEMS THAT IMPEDE EFFECTIVE PRODUCTION OF SOFTWARE-INTENSIVE SYSTEMS.

by Grady H. Campbell, Jr., Software Engineering Institute

The Role of Software in DoD Systems

As discussed in a recent issue (October 2007) of Software Tech News, software is a critical enabler of DoD system capabilities. As needed software capabilities grow in complexity, the challenges of building this software correctly, predictably, rapidly, and cost-effectively increase. The only hope for getting ahead of these challenges is to fundamentally improve the means and methods by which we build software. We have the opportunity to achieve substantial improvements over the next 10-20 years if we begin to invest systematically for software producibility.

Producibility is a comprehensive term for the ability to deliver needed capability in a timely, cost-effective, and predictable manner. OSD, as part of an exploratory effort with service Science and Technology representatives, has sponsored preliminary work toward establishing a Software-intensive Systems Producibility Initiative (SiSPI). The SiSPI is envisioned as a collaborative government-industry effort to create technology that will enhance producibility benefiting DoD acquisition and sustainment of software-intensive systems. SiSPI-associated work has included workshops with researchers and industry that have led to the development of a draft technology vision and roadmap [1]. The purpose of this roadmap is to frame and guide future investments toward making improved producibility a practical reality.

The need for producibility improvements has been recognized, lamented, and analyzed for largely the entire history of software engineering as a concept. Nevertheless, improvements in practice have occurred at only a modest rate even as DoD's dependence on software continues to grow (productivity improvements of perhaps 5-10 percent per year, with increasing size and complexity and undetermined effects on quality). True improvement requires technological advances that will provide the means for predictable, streamlined production of software and systems.

Producibility Problems in Practice

Over the last ten years there have been periodic attempts to characterize key software challenges and to advocate research efforts that could address them. With each such attempt, promising ideas emerge, often reiterating conclusions of past attempts, but all have failed to gain backing of decision makers for needed funding. Many government and industry executives seem confident that commercial approaches and routine incremental advances in software tools and methods will suffice to meet future DoD needs, precluding the need to invest further in research or development of such technology. Nevertheless, the costs, schedule uncertainties and delays, and product quality issues resulting with current production approaches remain high.

The SiSPI is an attempt to remedy this by addressing several aspects that past efforts have neglected: establishing a guiding vision and comprehensive evolving framework rather than focusing on specific technologies or approaches; focusing on improvements that address acquisition, production, and sustainment challenges rather than encompassing the infinite horizon of potential advances in end-use system capabilities; and measuring progress in terms of technologies actually adopted and showing benefit in practice rather than being satisfied with demonstrations of concept and hoping productization and adoption will occur naturally.

The primary motivation for SiSPI is to identify and solve the underlying problems that impede effective production of software-intensive systems. These problems pervade current acquisition and engineering approaches and are not likely to be solved with simple narrowly focused tools. Problems that the SiSPI work has identified as needing to be addressed to achieve producibility improvements, representative of most past such analyses, can be summarized briefly as:

- Requirements inadequately define the problem and over-specify a solution.
- Accommodations are not made in anticipation of likely future changes in requirements or technology.
- Testing consumes inordinate resources to find product flaws late.
- Projects ineffectively balance function, quality, and cost-schedule.
- Decisions made to achieve first delivery increase life cycle costs.
- System designs fail to explore software alternatives and tradeoffs.
- System properties due to software are not estimated and engineered but “fixed” through trial-and-error.
- Development tools do little to enhance development efforts, being useful only for recording as-built descriptions of a product.

Addressing all of these problems requires a comprehensive reconception of how we build and evolve software-intensive systems.

The Manufacturing of Software

These producibility problems are not inevitable or unavoidable; they are the symptoms of a flawed approach to the production and evolution of software-intensive systems. Current approaches to software and systems engineering trace to a time when software was a minor element of systems and needs were modestly conceived and easily understood. However, completed software was not easily changed so requirements had to be fixed early for an effort to succeed.

These are no longer the conditions under which software-intensive systems are built: software to a large degree today determines a system’s capabilities with hardware as its enabler: requirements are complex and changing, enabling hardware technologies and even the operational environment are all software-based and similarly complex and changing. The result is that the software defining the system must be regularly modified for the system to continue to meet the dependent enterprise’s changing needs.

To respond to this increasing complexity, a new vision is needed to describe our conception of how to properly build and evolve software and systems. This vision embodies a manufacturing concept, computer-aided design and manufacturing (CAD/CAM), not just for hardware elements but for software as well and for the system as a whole.

From a production perspective, systems engineers have always understood that hardware components had to be manufactured and that the cost and quality of the resulting product was dependent on the quality of the manufacturing capability used. A major phase of the acquisition process is concerned with determining how components will be manufactured, arranging supply of raw materials, and constructing and testing the needed manufacturing facilities.

Although the ease of identical replication of software suggests that manufacturing has no relevance to software, the mass customization approach to manufacturing offers a model that is highly applicable to software. This model has been realized and shown to work for software in the form of product line approaches. SiS I proposes extending this thinking to software production in general with the perspective that accommodating changing needs and technology, during both development and sustainment, are key challenges for software-intensive systems.

For software in particular, based on the perspective that most software produced by a qualified enterprise establishes and follows proven forms appropriate to the needs of that enterprise and its customers, formulating a manufacturing approach to software production based on those forms can enable the elimination of large parts of current acquisition, development, and sustainment efforts. Unfortunately but not surprisingly, current software development tools and methods do not conceive of nor provide appropriate support for such a CAD/CAM approach. This requires that we undertake a program of research, product development, and transition in order to make such an approach feasible for DoD acquisition and sustainment.

A Producibility Vision

From the roadmap, producibility has three dimensions that suggest the capabilities of SiS production that need to be addressed:

- Developer productivity: the efficiency and effectiveness of developers in creating and evolving a product
- Product value: the utility and quality of each product that results
- Acquirer acuity: the insight and foresight that acquirers have in delineating current and future capabilities needed

The producibility vision of CAD/CAM for software-intensive
systems (SiS) must address the production challenges that each of these perspectives presents.

Taking a broad view of the meaning of CAD/CAM in industry, CAD is the conception, design, and analysis of a problem and solution in model form while CAM is the manufacture from raw and processed materials of a product that conforms to that model. The roadmap identifies five principles that characterize this vision of CAD/CAM for SiS:

- Model-centric: All problem-solution information is expressed in a comprehensive multi-faceted model of a product and its envisioned context of use.
- Virtualized: A system is defined by building, pre-deploying, and validating in a software form within a hardware/software/user virtual environment.
- Predictable: Software and dependent system properties of interest are able to be accurately predicted and mutually optimized as a product model evolves.
- Decision-focused: Multiple alternative solutions are modeled, produced, and empirically evaluated based on identified customer and engineering decisions.
- Evolvable: The problem-solution is continuously evolved to create variant products that satisfy anticipated differing or changing needs.

The SiSPI technology roadmap provides a framework for identifying and initiating actions to create and transition producibility improvements into practice.

A Framework for Research

The roadmap identifies five areas of research focus required to achieve the producibility vision:

- Model-based development: Bridging the conceptual gap between customers and product developers to rapidly formulate, build, and evaluate alternative solutions to evolving needs
- Predictable software attributes: Measuring, predicting, and controlling SiS software properties and tradeoffs
- System virtualization: Creating virtualized environments for realistically evaluating alternative solutions
- Disciplined methods: Applying effective methods for engineering discipline in the development of software within systems
- Infrastructure and emerging technology: Exploiting changing infrastructure and computing technology capabilities for enhanced producibility

The first area derives directly from the vision whereas the second and third areas address the major impediments to achieving the full objectives of the first. The final two areas reflect the need to accommodate beneficial and inevitable advances in the practices and enabling technologies associated with software development independent of the producibility vision.

A Framework for Transition

Research advances alone do not result in producibility improvements in practice. Many seemingly sound results of research have failed to influence practice. The real challenge for DoD is to transition beneficial technology from research into effective practice. Starting with a proposed improvement based upon promising research results and existing practices, transition comprises three stages:

1. Validation – Evaluating proposed technology
2. Integration and productization – Preparing the technology for production use
3. Adoption – Instituting use of the technology on programs

Technology consists of both tools and methods and must be conceived to operate effectively within the context of other producibility-enhancing technologies, as well as any retained existing practices. These may also require revisions of existing practices and procedures to enable potential improvements. An objective of validation, integration, and productization is to minimize unnecessary disruptions to acquirers and developers,
but the objective of adoption is to make organizational changes that would otherwise impede successful use of the technologies that become available as a result.

**SiSPI Management Factors**

Based on a tentative governance approach that envisions a collaborative government-industry effort, the roadmap focuses on key aspects of initiative management that have the greatest implications for research and transition efforts:

- How potential research and transition efforts are to be identified and prioritized
- How effectiveness of research and transition efforts are to be measured

In light of realistic limits on potential funding and needed expertise, it is essential that the SiSPI target advances that provide near-term benefit while leading to efforts that achieve long-term advancement. The producibility vision provides an objective framework for judging potential contributions to long-term benefits but judging near-term benefits must rely on proper measurement of technology use in practice.

As near-term advances progress, interim visions of effective practice will be formulated to ensure that their adoption is compatible with current practices that are not as advanced. Evaluating the near-term benefit of narrowly focused technologies will require researchers to identify the measurable benefits they expect to achieve and that must be proven before becoming candidates for transition efforts leading to adoption.

A **Product Line Perspective on Producibility**

For each of the five categories of research identified, the roadmap begins to define goals whose attainment will provide the technological capabilities (tools and methods) needed to implement the producibility vision as a systematic approach for the production of software-intensive systems. In the interim, much of this vision can be implemented today within a product line context. While we may lack the generally applicable scientific insights to apply this vision to build an arbitrary system today, the limiting assumptions that underlie a product line offer a context in which more limited methods are sufficient.

The CAD/CAM vision of producibility revisits the original motivation for the product line concept, which was pragmatically limited to producing a set of similar products corresponding to the scope of a business enterprise [2]. This new vision encompasses product line products while also acknowledging the potential for other bases for limiting the scope of applicability of the production capability that results from domain engineering like activities. This particularly applies to capabilities that span business areas by providing solutions for broadly acknowledged needs with capabilities that are not themselves complete products but that serve as components of other business-directed application products. Similarly, this provides a framework for the development and evolution of one-of-a-kind and one-size-fits-all products that are long-lived and supportive of changing needs.

To realize this vision beyond the product line context still requires significant advances in our understanding of software as an artificial construct that must both correctly sense and represent the world in which it operates and also act effectively within it.

**Producibility problems are not inevitable or unavoidable; they are the symptoms of a flawed approach to the production and evolution of software-intensive systems.**

Once technologies are proven to have claimed benefits and have been engineered to a level of production quality, the emphasis will shift to reorienting acquisition programs toward rethinking their approach to acquisition, development, and sustainment, leading to the expedited adoption of appropriate enabling technologies. Changes in DoD acquisition policies and practices may be necessary to permit and encourage programs toward undertaking the effort of making such improvements.

It also goes beyond a narrow conception of product lines, that focuses only on software, to encompass systems engineering and customized hardware manufacturing as elements of a complete product that are interdependent and based on a shared view of customer needs and related engineering tradeoffs.

**Software Research Beyond Producibility**

The concept of producibility suggests a focus on DoD’s need to become more effective, in terms of cost, quality, and timeliness, in providing systems that support future operational capabilities. This can be expressed in the form of a DoD objective
Acquire improved capabilities, responsive to force objectives and needs, while adhering to schedule, budget, and quality goals.

When thinking about software research, it is natural to focus not on a goal such as this but rather on goals related to software advances that will enhance operational capabilities. The broad applicability of software means that potential research topics have few natural limits. The SiSPI effort focuses on producibility challenges because this has implications across all DoD efforts. If one instead considers specific DoD capability needs of the future, it is possible to characterize other high-level objectives that suggest many other potential areas for software research:

- Accurately observe and represent the evolving (past, present, and projected future) state of the world, exposing the relative quality and timeliness of available data, for more effective situation assessment and action planning.
- Obtain comparative predictions of alternative future operational states as projections of the estimated past and current state and potential actions as aids to action decisions and planning.
- Communicate and collaborate securely among cooperating forces as a virtual enterprise.
- Act over distance and time for predictable results.
- Scale operational capabilities to make most effective use of available resources under fault, failure, and overload conditions, in addition to nominal/routine, maintenance, and training conditions.

Objectives such as these suggest many potential directions for software research. Research into software capabilities supporting such objectives can be traced directly to DoD mission needs and progress against them can be judged from that perspective. Accordingly, other recent proposals have been made to pursue software-focused research that would better address objectives such as these. One concerning the challenges of future Ultra-Large-Scale Systems [3] and another concerning Cyber-Physical Systems that achieve more effective integration of computational and physical processes [4] are of particular interest. Proposals such as these anticipate advances in producibility as part of making their advances adoptable in practice, and the SiSPI envisions advancing such capabilities in collaboration with these efforts and others both in the U.S. and internationally.

References


About the Author

Grady Campbell, a member of the technical staff at the Software Engineering Institute since 2002, works to identify, develop, and transition improvements in the process and practices of software development and acquisition. Prior to this, Mr. Campbell was an independent consultant in software product lines after being responsible at the Software Productivity Consortium in 1990 for conceiving and developing the first comprehensive software product line methodology. His 35 years experience also includes participating in the NRL Software Cost Reduction project and leading development in the 1980’s of a model-based application generator based on adaptable software. He received a B.S. in Mathematics from Auburn University and an M.S. in Computer Science from the George Washington University.

Author Contact Information

Email: ghc@sei.cmu.edu

GSwERC is a set of recommendations for a master’s level graduate program in software engineering together with guidance to aid a university or other academic institution in implementing a GSwERC-based degree program.

by Richard Turner and Art Pyster, Stevens Institute of Technology

Over 50 universities in the United States and many others globally offer a Master’s Degree in Software Engineering (SwE). However, the most current SwE reference graduate curriculum was developed by the Software Engineering Institute at Carnegie Mellon over 15 years ago[1]. Given how differently today’s software is used and developed, a fresh look at graduate programs is needed. A broad coalition of professionals from academia, industry, and government is creating a new Graduate Software Engineering Reference Curriculum (GSwERC).

GSwERC is a set of recommendations for a master’s level graduate program in software engineering together with guidance to aid a university or other academic institution in implementing a GSwERC-based degree program. GSwERC defines requirements for a terminal professional master’s degree, analogous in many ways to a master’s of business administration. It includes outcomes, expected skills, knowledge and experience on entry, a body of knowledge, and an architectural framework, plus materials describing their creation, implementation, and evolution. GSwERC is envisioned as a living document that will be revisited regularly and updated when necessary to ensure relevance to the rapidly evolving software engineering discipline.

Table 1 – CAT Members and Observers

1. Rick Adcock, Cranfield University and INCOSE, UK
2. Edward Alef, General Motors, USA
3. Bruce Amato, Department of Defense, USA
4. Mark Ardis, Rochester Institute of Technology, USA
5. Larry Bernstein, Stevens Institute of Technology, USA
6. Barry Boehm, University of Southern California, USA
7. Pierre Bourque, Quebec University and SWEBOK volunteer, Canada
8. John Brackett, Boston University, USA
9. Murray Cantor, IBM, USA
10. Lillian Cassel, Villanova and ACM volunteer, USA
11. Robert Edison, Analytic Services Inc., USA
12. Richard Fairley, Colorado Technical University, USA
13. Dennis Frailey, Raytheon and Southern Methodist University, USA
14. Gary Hafen, Lockheed Martin and NDIA, USA
15. Thomas Hilburn, Embry-Riddle Aeronautical University, USA
16. Greg Hislop, Drexel University and IEEE Computer Society Educational Activities Board representative, USA
17. David Klappholz, Stevens Institute of Technology, USA
18. Philippe Kruchten, University of British Columbia, Canada
19. Phil Laplante, Pennsylvania State University, Great Valley, USA
20. Qiaoyun (Liz) Li, Wuhan University, China
21. John McDermid, University of York, UK
22. James McDonald, Monmouth University, USA
23. Ernest McDuffie, National Coordination Office for NITRD, USA
24. Bret Michael, Naval Postgraduate School, USA
25. William Milam, Ford, USA
26. Ken Nidiffer, Software Engineering Institute, USA
27. Art Pyster, Stevens Institute of Technology, USA
28. Mary Shaw, Carnegie Mellon University, USA
29. Sarah Sheard, Third Millenium Systems, USA
30. Robert Suritis, IBM, USA
31. Massood Towhidnejad, Embry-Riddle Aeronautical University, USA
32. Richard Thayer, California State University at Sacramento, USA
33. Barrie Thompson, University of Sunderland, UK
34. Richard Turner, Stevens Institute of Technology, USA
35. Joseph Urban, Texas Tech University, USA
36. Ricardo Valerdi, MIT & INCOSE, USA
37. David Weiss, Avaya, USA
38. Mary Jane Willshire, Colorado Technical University, USA
Development Approach

Stevens Institute of Technology began the project in Spring 2007 with sponsorship from the U.S. Department of Defense. In July 2007, Stevens formed a Curriculum Author Team (CAT) of invited experts from industry, government, academia, and professional associations. As GSwERC has matured, CAT participation has grown. Table 1 lists the current CAT.

The CAT met during four workshops in August 2007 (Arlington, Virginia), December 2007 (Arlington, Virginia), April 2008 (Hoboken, New Jersey), and July 2008 (Monterey, California). Discussions during and after the first two workshops led to the preparation and release of GSwERC 0.25, which was reviewed by more than 40 experts from around the world. The April and July workshops examined reviewer comments and planned out GSwERC 0.50, the current release, suitable for early adoption and broad review. Version 1.0 of the curriculum, incorporating community feedback, will be published in 2009.

GSwERC Guiding Principles, Assumptions, and Context

The following guidance was strongly influenced by the principles stated in SE2004\(^1\); in some cases, they repeat verbatim the wording of those principles. Differences arise primarily by distinguishing the higher expectations of graduate education from those of undergraduate education and by more explicitly recognizing how the software engineering discipline ties to other disciplines, notably systems engineering. The statements without elaboration are:

1. The principal purpose of GSwERC will be to provide set of tailorble recommendations for developing and improving curricula that provide software engineering education at the master's degree level.
2. The master's degree described by GSwERC will be a professional degree targeting practicing software engineers. With modification, GSwERC may serve as the foundation for those with a research interest who ultimately seek a doctoral degree; however, GSwERC is only designed to support professional degrees.
3. A master's program that satisfies GSwERC should require about as many credit hours as typical programs do now.
4. Software engineering is a distinct discipline with a rich body of knowledge, practice, and theory.
5. Software Engineering draws its foundations from a wide variety of disciplines, with deepening ties to systems engineering.
6. All software engineering students must learn to integrate theory and practice.
7. The rapid evolution and the professional nature of software engineering require an ongoing review and revision of the corresponding curriculum.
8. GSwERC will be sensitive to changes in technologies, practices, and applications, new developments in pedagogy, and the importance of lifelong learning.
9. GSwERC will go beyond knowledge elements to offer significant guidance in terms of individual curriculum components.
10. GSwERC will identify the fundamental skills and knowledge that all graduates of a SwE master's degree program must possess.
11. GSwERC will be based on a flexible curriculum architecture and on recognized bodies of knowledge. The latter will be modified and enhanced as required.
12. GSwERC will honor individual program and student flexibility by limiting the common knowledge required for all students to no more than 50% of the total knowledge taught in a master's program.
13. GSwERC will be broad-based and international in scope.
14. GSwERC will include exposure to aspects of professional practice as an integral component of the graduate curriculum.
15. GSwERC will include discussions of strategies and tactics for implementation, along with high-level recommendations.
16. The distinction between SE2004 and GSwERC will be clear and apparent.
17. GSwERC will identify expected knowledge and experience for students to enter a master's program in software engineering.

Expectations at Entry

Among the most challenging decisions is deciding what students should be capable of when they enter the master's program. After considerable discussion, the CAT agreed that students entering a masters program should have:

1. The equivalent of an undergraduate degree in computing or an undergraduate degree in an engineering or scientific field and a minor in computing.
2. The equivalent of an introductory course in software engineering, and
3. At least two years of practical experience in some aspect of software engineering or software development.

The rationale for these expectations is:

1. **Degree.** Many existing master’s programs in software engineering expect students to have a bachelor’s degree in an engineering or scientific field, but not a degree in computing. Such students generally bring much of the math skills and the ability to think analytically, both of which are essential to software engineering. Students often have programming experience, although it is usually programming in the small without the benefit of understanding how to address issues associated with large or complex software.

2. **Software Engineering.** The majority of master’s programs in a recent survey[3] do not start students in an introductory course in software engineering. These programs assume that the student has learned the equivalent knowledge either from earlier coursework or from professional experience. GSwERC follows the practice of the majority of programs in that regard.

Universities frequently offer an undergraduate course introducing software engineering to students who do not have the equivalent knowledge from a prior course or professional experience.

3. **Experience.** Software engineering is a practical field and it is a truism that there is no substitute for experience. The richness of the discussions in a graduate class and the sophistication of the analysis that a student can perform are driven, in part, by the experience of the students. Students with at least two years of practical experience in some aspects of software engineering or software development have a significantly deeper appreciation for the issues that are examined in the master’s program. Such experience should expose the student to a team environment and to working on several aspects of development as would happen when a student is part of a team modifying, testing, and releasing an existing application. The most germane experience would have the student (1) work on a component of a larger system that requires integration, (2) evolve an existing system; e.g., having to be backwards-compatible with previous versions; and (3) address contextual requirements of customers.

Developing a new SwE graduate curriculum is a daunting task, made more difficult by the rapid changes in the discipline of software engineering and the evolving relationship between software engineering and systems engineering.

In order to engineer software, a student must have mastered the fundamentals of computing, including programming, computer hardware, operating systems, data structures, algorithms, discrete math, and a design course that has considered developing a system in which a primary issue has been the integration of several components. Students who do not have at least a minor in computing will generally lack that mastery.

Universities frequently offer leveling courses to students who enter a master’s program lacking the expected background in computing.

2. **Software Engineering**. The majority of master’s programs in a recent survey[3] do not start students in an introductory course in software engineering. These programs assume that the student has learned the equivalent knowledge either from earlier coursework or from professional experience. GSwERC follows the practice of the majority of programs in that regard.

Universities frequently offer an undergraduate course introducing software engineering to students who do not have the equivalent knowledge from a prior course or professional experience.

3. **Experience.** Software engineering is a practical field and it is a truism that there is no substitute for experience. The richness of the discussions in a graduate class and the sophistication of the analysis that a student can perform are driven, in part, by the experience of the students. Students with at least two years of practical experience in some aspects of software engineering or software development have a significantly deeper appreciation for the issues that are examined in the master’s program. Such experience should expose the student to a team environment and to working on several aspects of development as would happen when a student is part of a team modifying, testing, and releasing an existing application. The most germane experience would have the student (1) work on a component of a larger system that requires integration, (2) evolve an existing system; e.g., having to be backwards-compatible with previous versions; and (3) address contextual requirements of customers.

Expectations at Graduation

Analogous to the statements about expectations at program entry, there are statements about what a student should be capable of at graduation. Currently, there are ten statements plus elaboration. The statements without elaboration follow.

Graduates of a master’s program that satisfies GSwERC recommendations will:
1. Have mastered the Core Body of Knowledge (CBOK).
2. Have mastered at least one application domain, such as finance, medical, transportation, or telecommunications, and have mastered one application type, such as real-time, embedded, safety-critical, or highly distributed systems. That mastery includes understanding how differences in domain and type manifest themselves in both the software itself and in their engineering, and includes understanding how to learn a new application domain or type.
3. Have mastered at least one knowledge area or sub-area from the CBOK to at least the Bloom Synthesis level.
4. Have demonstrated how to make ethical professional decisions and practice ethical professional behavior.
5. Understand the relationship between software engineering
and systems engineering and be able to apply systems engineering principles and practices in the engineering of software.

6. Be able to work effectively as part of a team, including teams that may be international and geographically distributed, to develop quality software artifacts, and to lead in one area of project development, such as project management, requirements analysis, architecture, construction, or quality assurance.

7. Show ability to reconcile conflicting project objectives, finding acceptable compromises within limitations of cost, time, knowledge, existing systems, and organizations.

8. Understand and appreciate the importance of feasibility analysis, negotiation, effective work habits, leadership, and good communication with stakeholders in a typical software development environment.

9. Understand how to learn new models, techniques, and technologies as they emerge, and appreciate the necessity of such continuing professional development.

10. Be able to analyze a current significant software technology, articulate its strengths and weaknesses, and specify and promote improvements or extensions to that technology.

Curriculum Architecture

The curriculum architecture is similar to the one included in the 1991 SEI report[1] and is compatible with the existing masters programs that were surveyed. It provides a structural basis for programs to package courses that will teach the body of knowledge and will enable students who meet the entrance expectations to ultimately satisfy the graduation expectations. Figure 1 provides an overview of the architecture.

Core Body of Knowledge

The most difficult task in the entire curriculum effort has been creating the Core Body of Knowledge (CBOK) – deciding what core knowledge is needed for a software engineer at the master’s level. If the core knowledge is too large, universities will not have the flexibility needed to tailor their programs. If the core knowledge is too small, the current fragmentation of graduate education will continue and the model curriculum will provide little value.

The primary source for the CBOK was the SWEBOK[4]. Additional knowledge elements were added from the SEEK[5] and the INCOSE BOK[7]. In the study and analysis of these sources, various changes were made to satisfy the GSwERC expected student outcomes and to accommodate the needs and views of industry, academia, and the computing professional societies.

Comparisons and Hypothetical Implementations

If GSwERC is to be useful, it must support its own implementation. To this end, GSwERC includes detailed comparisons with existing master’s programs as well as examples of how to evolve a curriculum so that it is consistent with the CBOK and architecture in a way that satisfies the expected outcomes.

Conclusions

Developing a new SwE graduate curriculum is a daunting task, made more difficult by the rapid changes in the discipline of software engineering and the evolving relationship between software engineering and systems engineering. The GSwERC
effort has had the benefit of a broad, dedicated and extremely capable author team and solid foundation documents on which to base the effort.

The version number 0.5 reflects over a year of work. We believe the guiding principles, expectations when students enter the program, and expected outcomes when students graduate are largely correct, although, of course, review by others will test that belief. The curriculum architecture is quite flexible and seems to support the variety of graduate programs that were studied at the beginning of this effort. The CBOK, on which the curriculum is based, is grounded in the SWEBOK with a modest number of additions and changes. Comparison with existing program seems to indicate an appropriate scope and depth in the CBOK and outcomes.

While certainly in need of broad validation and refinement, this version of GSwERC should serve as a foundation for the wider community to mature and as guidance for early adopters. ‘GSwERC is available for download at www.GSwERC.org. If you are interested in participating in the further development of the curriculum, please contact one of the authors.

References


About the Authors

Dr. Arthur Pyster is a Distinguished Research Professor in the School of Systems and Enterprises at Stevens Institute of Technology, Deputy Executive Director of the Systems Engineering Research-University Affiliated Research Center sponsored by the Department of Defense, and a member of the Board of Directors of the International Council on Systems Engineering. He has more than thirty years of experience as a successful senior executive, researcher, engineer, educator, and program and project manager in government, industry, and academia. He has created, delivered, and operated numerous leading edge systems and technologies in telecommunications, aerospace, defense, air traffic control, and information technology domains. Before joining Stevens in March 2007, Dr. Pyster served in several executive positions, including Senior Vice President and Director of Systems Engineering and Integration for SAIC and Deputy Chief Information Officer for the Federal Aviation Administration.

Author Contact Information

Email: Art.Pyster@stevens.edu

Dr. Richard Turner is a Distinguished Service Professor at Stevens Institute, a Visiting Scientist at the Software Engineering Institute of Carnegie Mellon University and a respected researcher and consultant with thirty years of international experience in systems, software and acquisition engineering. Dr. Turner has supported defense, intelligence and civil government agencies. A charter member of the author team for CMMI, he has led process improvement initiatives across a broad range of disciplines. Dr. Turner is co-author of three books: Balancing Agility and Discipline: A Guide for the Perplexed, CMMI Distilled, and CMMI Survival Guide: Just Enough Process Improvement.

Author Contact Information

Email: Richard.Turner@stevens.edu
Technical Training On Demand

Cost effective way for organizations to provide continuous learning opportunities for their employees

- Pay one annual subscription fee ($475)
- Take as many classes as you want

Classes available
24/7/52

From:
Home
Office
Travel

DACS On-Line Learning Center

450+ classes
12,000 topics
Programming + Web Development
Course Catalog

Latest Technologies
Java  XML
Oracle  Server Technologies
.NET Framework

To view the catalog visit
www.thedacs.com/training
For details call
1.800.214.7921

AFFORDABLE

- Fit course work into your schedule
- Self paced
- Refresh knowledge
- Study during lunch or after work

FLEXIBLE

COMPREHENSIVE

ACCREDITATION CERTIFICATION

ORACLE
- Certifications
- MCAD
- MCDRA
- MCTP
- MCPD
- MCSD
- MCSA
- MCSE
- MCTS
- MCAS

MICROSOFT
- Certification Suite
- CEUs granted
- IACET
- Accreditation Supported

Data & Analysis Center for Software (DACS)
Improvement at SAIC Defense Solutions Group

DSG believed that CMMI would help it establish improvement plans to achieve its business goals.

by Sharon Cobb Flanagan (SAIC) and Dennis R. Goldenson (SEI)

SAIC believes in continuously improving processes that add value to the products and services it provides to customers. SAIC’s Defense Solutions Group (DSG) (formerly System & Network Solutions Group) has used the Capability Maturity Model® (CMM®) for software and the Capability Maturity Model Integration (CMMI®), as well as other models and standards, in its improvement initiatives. Activities and quantifiable trends observed while transitioning from using the CMM-SW to using the CMMI in support of improvements were captured and are reflected within this article. Through years of using the CMM-SW and CMMI, the trends show a decrease in the average cost per person for process improvement while, during the same time periods, there are increases in employee training accomplishments, improvements in process performance (e.g., the peer review process and defect discovery), and increases in overall average customer satisfaction ratings.

The Organization and its Products

With a long history of reliable service to government and commercial organizations, DSG has expertise serving several markets. Vertical markets for the public sector include the federal, state and local government; vertical markets for the private sector include the technology industry, the telecommunications industry and transportation (automotive).

Primary emphasis in DSG is on system/software engineering, integration and life cycle support for complex command, control, communications, computers, intelligence, surveillance and reconnaissance (C4ISR) systems. Expertise, capabilities, and talent across DSG are brought together to do this. DSG integrates and pulls together multiple skill sets and capabilities across boundaries (business-based, functional and geographical) to pursue and perform on multi-disciplinary projects.

DSG provides customer-based services focused on customer satisfaction. As such, employees are geographically dispersed across approximately 30 locations, with the vast majority at five locations: San Diego, Calif., and McLean/Reston/Vienna, Va., Orlando, Fla., Crane, Ind., Middleton, R.I., Charleston, S.C., and Huntsville, Ala. There were approximately 4,300 employees within DSG in 2004 (FY2005 and the basis for the data in this report); they worked in various capacities on numerous contracts,
most of which provided domain-specific customer-focused solutions and services, such as research support, white papers, site support, program office support, engineering support, subcontract support, integrated product team support, systems engineering and integration support, etc. The organization’s annual average software product sizes range from 24.8 thousand source lines of code (KSLOC) to 107.1 KSLOC.

**Process Improvement History**

Known as SAIC’s Information and Technology Systems Sector (ITSS) until 2003, DSG has pursued process improvements since the early 90s. The primary focus within various groups throughout the organization was on using the CMM for software. In 2001, process improvement initiatives became group-wide, using both the SW-CMM and International Organization for Standardization (ISO) 9001:2000. DSG achieved SW-CMM maturity level 3 (validated by an external lead assessor), and ISO 9000 registration occurred in a key system engineering portion of the organization. SW-CMM maturity level 4 then was established in a key software development portion of the organization, followed by ISO 9001:2000 registration for that same part of the organization.

Transition to CMMI began in 2002. In addition to multiple internal assessments, audits, and quick look appraisals, externally appraised benchmarks included:

- DSG: CMMI SE/SW Maturity Level 5, December 2004
- DSG/C4IT/Advanced Systems Development & Integration Operation Omaha location: ISO 9001 registration, July 2004
- DSG (formerly SNSG & ITSS): CMMI SE/SW Maturity Level 3, December 2003
- DSG/FEDCOM: ISO 9001 registration, October 2003
- DSG/C4IT/ASDI Operation: ISO 9001 registered in January 2003
- DSG/C4IT/ASDI Operation: SW-CMM Maturity Level 4, January 2002
- DSG (formerly SNSG and ITSS): SW-CMM Maturity Level 3, November 2001

**CMMI-Based Process Improvement**

In addition to establishing management and engineering baselines for both systems and software using CMMI maturity level 3, DSG business plan goals that were part of overall continuous improvements for FY04 (calendar year 2003) included maintaining the current software maturity baseline while integrating maturity level 3 processes into new organizations resulting from reorganizations and acquisitions. Other FY04 goals related to the DSG culture of continuously improving included evolving ISO registrations; strengthening the measurement program; optimizing the effectiveness of the training program; and improving internal communications. The FY05 business plan goals included deploying innovative technologies that improve program performance; building upon program management processes; improving project review processes to incorporate solid technical review processes; ensuring scalable management, system engineering, and software development processes; and building up the organization’s system engineering base.

DSG believed that CMMI would help it establish improvement plans to achieve its business goals. DSG also had experienced performance benefits resulting from its use of the SW-CMM and expected to continue to see benefits from CMMI-based practices.

To strengthen its measurement program, DSG used best practices from Organizational Innovation and Deployment (OID) along with Organizational Process Performance (OPP), Quantitative Project Management (QPM) and Measurement and Analysis (MA). In so doing, DSG strengthened the measurement program plan it had used to support its activities based on SW-CMM maturity level 4.

Two internal tools that influenced improvements in project execution were developed as the organization improved its verification processes: a Microsoft Excel® tool that models DSG peer review process performance (VER) and an Excel-based tool used to predict software defects and to monitor defect discovery. The tools were initially calibrated based on historical data from the SAIC Organizational Metrics Data Base (OMDB) specific to DSG process performance that had been captured over several years. Using best practices of Organizational Process Focus (OPF), Organizational Process Definition (OPD), and Causal Analysis and Resolution (CAR), the organization established Process Action Teams (PATs) and conducted causal analyses of the tools’ performance against actual project data. The tools were recalibrated, observed on projects, and then deployed across DSG for use by projects and line management during project reviews. The peer review process performance supported the focus on improving the peer review process, since it was identified as important in meeting fiscal year financial and quality business goals as well as project performance goals.

To make the transition from the SW-CMM to CMMI, DSG
used best practices from OID, OPF, and OPD to review systems engineering processes existing within SAIC. Those processes were integrated into the DSG baseline software and management processes, while maintaining a focus on scalability. An internal Web-based tool was established to assist with identifying which processes were appropriate to all DSG projects group-wide (e.g., systems engineering, software, services, and network management). The tool also is used for documenting approved waivers based on defined criteria and for process tailoring.

While integrating the system and software engineering processes, the organization updated its project management processes to reflect project lessons learned and best practices from Project Planning (PP), Project Monitoring and Control (PMC), Integrated Project Management (IPM), QPM, Decision Analysis and Resolution (DAR), and CAR. DSG also strengthened its already existing process for analyzing measured project performance trends and for taking corrective action with respect to technical, cost, schedule, contract performance and customer satisfaction.

Through years of using the CMM-SW and CMMI, the trends show a decrease in the average cost per person for process improvement while, during the same time periods, there are increases in employee training accomplishments, improvements in process performance (e.g., the peer review process and defect discovery), and increases in overall average customer satisfaction ratings.

Continuously improving program management meant building upon the organization’s maturity level 3 project management processes. DSG integrated its improved measurement process, tools, and training for project management use in managing projects. The new defect prediction and peer review process performance tools assisted projects with proactively determining whether the peer review processes being applied on their projects were performing as planned (within the organization’s defined control limits). In cases where the tools indicated it might not have been, project team members looked for potential root causes and any opportunities for improvement, where appropriate. In one case, refresher training was reinforced; in another case, the standard definition of defect criticalities was clarified. In another case where the defect discovery rates did not appear to align with the phases where the quantity of defects was expected to be found, an DSG PAT was pulled together with project representatives from across the organization to look for potential root causes (OPF, CAR, and OPF). It was discovered that the tool was calibrated for full-scale development and needed a separate calibration for maintenance-related activities.

To assist with optimizing the effectiveness of the training program, DSG used processes based on OID to identify and deploy a Web-based training management tool. This helped with efforts to baseline the employee skill sets to include program management and system engineering. It also helped with DSG fiscal year planning and project planning to train employees based on needs related to the project roles.

DSG also decided to improve the application of its peer review process with a focus towards continuously improving upon product quality. This was accomplished by proactively looking for any defects earlier in the development process, where less effort is required to make corrections. This involved monitoring for potential trends in any product discrepancies discovered during peer reviews and testing, and conducting root cause analysis sessions on any identified (OID, OPP, QPM, VER, and CAR).

While SAIC previously had formal processes in place to aid in the making of major decisions, DSG used DAR to pull them together into a single tool, using a table that identifies which formal processes best apply for various types of decisions, including hot links to each formal process.

All of the organization’s more than 200 process assets reside on a Web site that is easily accessible to employees and projects at any time. The assets include policies, manuals, procedures, templates, checklists, forms, tools, and training materials. Best-in-class examples are available there as well.

**Performance Results**

DSG has experienced many benefits in performance along its process improvement journey. The organization does not quantify return on investment, per se; however, benefits are visible in many trends across the organization, reflecting an
The DACS Gold Practice Initiative:

- Promotes effective selection/use of software acquisition & development practices
- Defines essential activities/benefits of each practice
- Considers the environment in which each practice is used
- Addresses the timeliness of practice benefits
- Recognizes interrelationships between practices that influence success or failure
- Contains quantitative and qualitative information
- Includes discussion forums, case studies and survey opportunities
- A continually evolving resource for the DoD, Government, Industry and Academia
- Free to use/free to join

Current Gold Practices:

- Acquisition Process Improvement
- Architecture-First Approach
- Assess Reuse Risks and Costs
- Binary Quality Gates at the Inch-Pebble Level
- Commercial Specifications and Standards/Open Systems
- Ensure interoperability
- Formal inspections
- Formal Risk Management
- Goal-Question-Metric Approach
- Integrated Product and Process Development
- Model-Based Testing
- Plan for Technology Insertion
- Requirements Management
- Requirements Trade-Off/Negotiations
- Statistical Process Control
- Track Earned Value

Learn More About the DACS Gold Practice Initiative:
http://www.goldpractices.com

Become a Registered Member!
http://www.goldpractices.com/registration/register_step1.php

Data & Analysis Center for Software
P.O.Box 1400
Rome, NY 13442-1400

http://iac.dtic.mil/dacs
increased business value overall.

The average cost per person per year for process improvement expenditures decreased almost 10 percent during the transition from SW-CMM maturity level 3 in FY02 to CMMI maturity level 3 in FY04; costs decreased by almost 40 percent by the time the organization achieved CMMI maturity level 5 in FY05 (Figure 1). The marked drop in fiscal year 2003, after achieving SW-CMM maturity level 3, was because the organization was sustaining its then existing processes while establishing its initial plans to make the transition to CMMI. The relative rise in the following year reflects the activities to upgrade the baseline management and engineering practices to incorporate systems engineering practices and address providing services; to establish Web-based tools and Web portals to support the growing organization; to upgrade the management practices; and to establish and upgrade training related to the new and modified processes based on CMMI. Despite these considerable startup activities, DSG process improvement expenditures still declined from what they had been at a comparable time two years earlier.

As also seen in Figure 1, the organization’s process improvement expenditures again dropped markedly the following fiscal year, when DSG achieved CMMI maturity level 5. The organization was able to build on the upgrades that were put in place for CMMI maturity level 3. DSG was already looking ahead to maturity level 5 from the beginning of its move to CMMI, and its existing measurement program plan based on SW-CMM level 4 practices already had proven to be effective in one part of the larger organization. Some of the decline in process improvement expenditures in FY05 was made possible by tool investments made the previous year to support the causal analysis practices that the organization had begun to put in place earlier.

Completion of training, overall, continued to increase from SW-CMM maturity level 3 through CMMI maturity levels 3 and 5 (Figure 2). This included management and technical training as well as strictly process related training. There was a slight drop in process training completions after DSG achieved SW-CMM maturity level 3 because the majority of the process training already had been accomplished. The amount of training delivered increased again in FY04 as the organization moved to CMMI maturity level 3, primarily because of the need to train employees new to the organization due to reorganizations and acquisitions. New process training requirements also emerged to support the organization’s transition to CMMI-based processes, for example, for systems engineering, services, the DSG measurement program, conducting causal analysis, and new tool usage.

Peer reviews have become more efficient as a result of CMMI-based improvement at DSG (Figure 3). Defects are being found earlier, when it costs less to fix them, and there are now fewer defects to find. The marked increase is primarily due to the institutionalization of peer reviews across many types of products in addition to software, a process change that was reinforced as a result of observing benefits from the organization’s earlier practices based on the SW-CMM. Similarly, labor productivity as measured by source lines of code produced per person per period increased by 59 percent during the same period of time.

As seen in Figure 4, the number of major defects found prior to release dropped markedly over the four-year period. Following a noticeable rise in FY03 from the SW-CMM baseline, by the time the organization achieved CMMI maturity level 5, it found
fewer than a quarter as many defects than it had found originally. Similarly, the number of major defects being discovered during system testing at CMMI maturity level 3 dropped to 72 percent of what had been reported at SW-CMM maturity level 3. The number of major defects found during system testing dropped to 28 percent of the SW-CMM baseline at CMMI maturity level 5. Both improvements appear to be due to the improved effectiveness of the peer review process to find defects earlier.

As also seen in Figure 4, overall product quality increased as measured by major defects discovered post-release at CMMI maturity level 3, which had dropped to 19 percent of what had been reported at SW-CMM maturity level 3. The figure does not contain a comparable data point for CMMI maturity level 5 because very little post release data are available. Although relatively few defects appear to have escaped, to be found post release, the organization continues to analyze and track these figures.

A steady increase in overall customer satisfaction occurred as DSG progressed from SW-CMM maturity level 3 to CMMI maturity level 5. The organization’s customer satisfaction ratings, on average, already were high, and they continued to increase (Figure 5). The ratings are measured on a scale of 1-10, where the clients select numerical values reflecting their overall degree of satisfaction with the work performed. Feedback is solicited multiple times over the life of each customer engagement.

As the organization moved from SW-CMM maturity level 3 to CMMI maturity level 5, customer satisfaction ratings became an important element of the organization's measurement program. Additional emphasis was given to ensure institutionalization and timely feedback to customers. The organization's operating units established goals higher than their previous average ratings to ensure a focus on improving customer satisfaction. Hence, the slight drop in the trend in FY05 may be due to the average rating being a function of more client assessments being conducted that year.

Overall, DSG has accomplished more for less cost during its journey to CMMI maturity level 5. Projects have come in under planned budget. Overall process improvement expenditures have declined even though there has been a steady increase in training costs as the group’s maturity has improved. More defects are being found earlier and productivity has improved markedly. Measured customer satisfaction has improved along with a decline in defect density, both pre-release and post release.

Figure 3: Change since FY02 in Defects Found per Peer Review Hour

Figure 4: Change since FY02 in Major Defect Rates Pre- and Post-Release

Figure 5: Change since FY02 in Customer Satisfaction

Capability Maturity Model, CMM and CMMI are registered trademarks of Carnegie Mellon University in the United States. ISO is a registered trademark of the International Organization for Standardization in the United States and/or other countries. Microsoft and Excel are registered trademarks of Microsoft Corporation in the United States and/or other countries.
About the Authors
Sharon Cobb Flanagan, Senior Vice President in the Science Applications International Corporation (SAIC), serves as Director of Quality Assurance and Process Improvement for the Defense Solutions Group. She has over 20 years of experience in product development, management, quality assurance, process improvement, and research. She is a member of the Software Engineering Institute (SEI) program, American Society for Quality, and IEEE computer Society, and is an SEI-certified Lead Appraiser and Instructor and Six Sigma Black Belt. She holds a bachelor’s degree in computer systems from Rochester Institute of Technology, and has completed courses towards a master’s degree in organizational behavior.

Author Contact Information
Email: sharon.cobb.flanagan@saic.com

Dennis R. Goldenson is a senior member of the technical staff at the Software Engineering Institute. Dr. Goldenson came to the SEI in 1990 after teaching at Carnegie Mellon University since 1982. A principal author of the CMMI Measurement and Analysis process area, he has served both as technical lead for the SEI’s empirical investigations of the performance outcomes of CMMI and as international trials coordinator for the SPICE project in support of ISO/IEC 15504. Goldenson has published numerous papers and made many presentations on improving measurement practice; the improvement of process models and appraisal methods; and the impact and transition of process improvement and other engineering technologies. Related interests are in survey research, experimental design, the visual display of quantitative information, the quantitative analysis of textual information, and tools to support collaborative processes.

Author Contact Information
Email: drg@sei.cmu.edu

Recent Technical Reports from DACS

  Released: 9/30/2007

- Enhancing the Development Life Cycle to Produce Secure Software
  Released: Version 2.0, October 2008
  https://www.thedacs.com/techs/enhanced_life_cycles/

  Released: August 2007

  Released: 9/30/2007

PDF versions of these reports can be downloaded free of charge by registered DACS visitors who are logged in.

Hard copies can be ordered (for fee) from the DACS store online at https://store.thedacs.com/
Are you getting **All** you can from your software investment?

The DACS ROI Dashboard

**Impact on Quality (% defect reduction)**

75th Percentile: 90

Oklahoma City Air Logistics Center, Test Program Set and Industrial Automation line observed the defect density decrease 91%, from 3.3 to 0.3 Defects per KLOC, while achieving CMM Level 4.

Access the DACS ROI Dashboard!

http://www.thedacs.com/databases/roi/

**Technologies Covered:**
- SEI CMM/CMMI
- SEI Team Software Process (TSP)
- SEI Personal Software Process (PSP)
- Inspections
- Reuse
- Cleanroom
And Many More!

**Graphs Showing Impact Of Software Technologies On:**
- ROI
- Productivity
- Quality

**Summarizes Facts From Open Literature**

Data & Analysis Center for Software
P.O.Box 1400
Rome, NY 13442-1400

http://iac.dtic.mil/dacs
STN Article Submission Policy

The STN is a theme-based quarterly journal. In the past DACS has typically solicited specific authors to participate in developing each theme, but we recognize that it is not possible for us to know about all the experts, programs, and work being done and we may be missing some important contributions. Therefore, beginning in 2009 DACS is adopting a policy of accepting articles submitted by the software professional community for consideration.

DACS will review articles and assist candidate authors in creating the final draft if the article is selected for publication. Note that DACS does not pay for articles published. Note also that submittal of an article constitutes a transfer of ownership from the author to DACS with DACS holding the copyright.

Although the STN is theme-based, we do not limit the content of the issue strictly to that theme. If you submit an article that DACS deems to be worthy of sharing with the community, DACS will find a way to get it published. However, we cannot guarantee publication within a fixed time frame in that situation. Consult the theme selection page and the Author Guidelines located on the STN web site (see https://www.softwaretechnews.com/) for further details.

To submit material (or ask questions) contact news-editor@dacs.dtic.mil

Tentative Themes for 2009 are:

- Earned Value
- Software Testing
- Project Management
- Model Driven Development

Announcing a Correction!


The author has informed DACS that in his article, Figure 3 was attributed to Herb Krasner who used it in a presentation, but it should have been attributed to Capers Jones who actually created it. The original source is:

Jones, Capers; “Software Quality - a Survey of the State of the Art”; Software Productivity Research LLC, Narragansett, RI; 2008

DACS is publishing this correction because the printed version of the STN had already been distributed when DACS was informed of the correction. DACS has subsequently changed the online version of this article to appropriately credit Mr. Jones as the originator and Mr. Krasner as a user of the information.
The first 50 people to send in a completed survey will receive a FREE DoD/IT Acronym CD from the DACS.

This valuable CD-ROM contains over 9,000 Department of Defense and Information Technology acronyms. There are hundreds of acronym lists available but none are as well done as this CD AND specifically targeted towards DoD and Information Technology. This unique-shaped CD-ROM plays in your computer's regular, hub-mounted, CD drive. You'll use this great resource over and over again. It's FREE, just for filling out our brief survey on the next page!

Data & Analysis Center for Software (DACS)

http://iac.dtic.mil/dacs/

Data & Analysis Center for Software (DACS)

775 DIACALIAN DR
ROME NY - 3449-0193
1. Which volume of the Software Tech News did you receive? __________________________________________

2. When did you receive the newsletter? (month/year) _______________________________________________

3. How satisfied were you with the CONTENT of the newsletter? (Article Quality)
   - Very Satisfied
   - Satisfied
   - Neither Satisfied nor Dissatisfied
   - Dissatisfied
   - Very Dissatisfied

4. How satisfied were you with the APPEARANCE of the newsletter?
   - Very Satisfied
   - Satisfied
   - Neither Satisfied nor Dissatisfied
   - Dissatisfied
   - Very Dissatisfied

5. How satisfied were you with the OVERALL QUALITY of the newsletter?
   - Very Satisfied
   - Satisfied
   - Neither Satisfied nor Dissatisfied
   - Dissatisfied
   - Very Dissatisfied

6. How satisfied were you with the ACCURACY of the address on the newsletter?
   - Very Satisfied
   - Satisfied
   - Neither Satisfied nor Dissatisfied
   - Dissatisfied
   - Very Dissatisfied

7. Approximately how much of the newsletter do you read?
   - The entire issue
   - Most of the content
   - About half the content
   - Briefly Skimmed
   - Didn’t Read

8. Would you read this newsletter in an E-mail newsletter format?
   - Definitely
   - Probably
   - Not Sure
   - Probably Not
   - Definitely Not

9. How did you request the product or service?
   - Phone Call
   - E-mail
   - DACS Website
   - Subscription Form
   - Other ______________________

10. Would you recommend the Software Tech News to a colleague?
    - Definitely
    - Probably
    - Not Sure
    - Probably Not
    - Definitely Not

11. What topics would you like to see this newsletter devoted to? ______________________________________

Comments (optional) _______________________________________________________________________
_____________________________________________________________________________________

REGISTER FOR THE FIRST TIME | UPDATE CURRENT SUBSCRIPTION

Name: ___________________________________________ Position/Title: __________________________

Organization: __________________________________ Organization Type: ______

Address: ______________________________________ City: _________________________________

State: __________________ Zip: ______________ Country: _________________________________

Telephone: ______ - ______ - ______ Fax: ______ - ______ - ______ Email: _______________________

Functional Role: _______________________________________________________________________

Organization Type: □ Air Force □ Army □ Navy □ Other DoD __________________________
□ Commercial □ Non-Profit □ Non-US □ US Government □ FFR&D □ Other _________________

*Note: You must give us your address to receive the CD.
The Software Tech News is published quarterly by the Data & Analysis Center for Software (DACS). The DACS is a DoD sponsored Information Analysis Center (IAC), administratively managed by the Defense Technical Information Center (DTIC). The DACS is technically managed by Air Force Research Laboratory, Rome, NY and operated by ITT, Advanced Engineering and Sciences Division.

ARTICLE REPRODUCTION

Images and information presented in these articles may be reproduced as long as the following message is noted:

“This article was originally published in the Software Tech News, Vol. 11, No. 4 December 2008. Requests for copies of the referenced newsletter may be submitted to the following address:

Philip King, Production Editor
Data & Analysis Center for Software
P.O. Box 1400
Rome, NY 13442-1400

Phone: 800-214-7921
Fax: 315-838-7130
E-mail: news-editor@dacs.dtic.mil

In addition to this print message, we ask that you notify DACS regarding any document that references any article appearing in the Software Tech News.

ABOUT THIS PUBLICATION

The Software Tech News is published quarterly by the Data & Analysis Center for Software (DACS). The DACS is a DoD sponsored Information Analysis Center (IAC), administratively managed by the Defense Technical Information Center (DTIC). The DACS is technically managed by Air Force Research Laboratory, Rome, NY and operated by ITT, Advanced Engineering and Sciences Division.
IN THIS ISSUE

Tech Views — DoDTechipedia for Software and IT Professionals
by Thomas McGibbon, DACS Director, ITT Corporation ................................................................. 3-4

Software Technologies in Technology Readiness Assessments for DoD Acquisition Programs
by Robert Gold, Deputy Under Secretary of Defense for Science & Technology DUSD (S&T) .................. 5-7

Ultra-Large-Scale Systems: Challenges and Promising Research Areas
by Linda M. Northrop, Software Engineering Institute ................................................................. 9-12

Advancing Producibility for Software-Intensive Systems
by Grady H. Campbell, Jr., Software Engineering Institute ............................................................. 13-17

The Graduate Software Engineering Reference Curriculum: A Joint Industry, Government and
Academic Project
by Richard Turner and Art Pyster, Stevens Institute of Technology .................................................... 18-22

Improvement at SAIC Defense Solutions Group
by Sharon Cobb Flanagan (SAIC) and Dennis R. Goldenson (SEI) ..................................................... 24-30