

# CROSSTALK

July 2004    **The Journal of Defense Software Engineering**    Vol. 17 No. 7



## Top Five Quality Software Projects

### 4 Winning Projects Exemplify Success for Developers and Acquirers

CROSSTALK is proud to present this series of articles featuring the winners of the 2003 U.S. Government's Top 5 Quality Software Projects.

by Elizabeth Starrett

5  
Top

- 6 The Advanced Field Artillery Tactical Data System Proves Successful in Battle
- 8 The DMLSS Program Brings Electronic Commerce to the Military Medical Treatment Facilities
- 10 The H1E System Configuration Set Lays the Foundation for Decades to Come
- 12 The OneSAF Objective System Fits Individual Simulation Needs
- 14 Patriot Excalibur Software Enables Full-Scale Deployment of Battle-Ready Units
- 18 CROSSTALK Honors the 2003 Top 5 Quality Software Projects Finalists



**ON THE COVER**

Cover Design by  
Kent Bingham.

## Software Engineering Technology

### 20 Using Software Metrics and Program Slicing for Refactoring

These authors discuss how program slices produced from a single software module are sorted by the respective values of the metrics to guide refactoring, which improves software system quality.

by Dr. Ricky E. Sward, Dr. A.T. Chamillard, and Dr. David A. Cook

### 25 Right Sizing Quality Assurance

This article introduces quality efficiency indicators that facilitate right sizing the quality assurance function to the customer's need, or the producer organization's own quality goals.

by Walt Lipke

## Departments

- 3 From the Publisher
- 16 Top 5 Awards Presentation Ceremony
- 24 Coming Events
- 28 SSTC 2004 Conference Highlights
- 30 Web Sites  
Call for Articles
- 31 BACKTALK

## CROSSTALK

PUBLISHER	Tracy Stauder
ASSOCIATE PUBLISHER	Elizabeth Starrett
MANAGING EDITOR	Pamela Palmer
ASSOCIATE EDITOR	Chelene Fortier-Lozancich
ARTICLE COORDINATOR	Nicole Kentta
CREATIVE SERVICES COORDINATOR	Janna Kay Jensen
PHONE	(801) 586-0095
FAX	(801) 777-8069
E-MAIL	crosstalk.staff@hill.af.mil
CROSSTALK ONLINE	www.stsc.hill.af.mil/ crosstalk

**Subscriptions:** Send correspondence concerning subscriptions and changes of address to the following address. You may e-mail or use the form on p. 30.

Ogden ALC/MASE  
6022 Fir AVE  
BLDG 1238  
Hill AFB, UT 84056-5820

**Article Submissions:** We welcome articles of interest to the defense software community. Articles must be approved by the CROSSTALK editorial board prior to publication. Please follow the Author Guidelines, available at <www.stsc.hill.af.mil/crosstalk/xtlguid.pdf>. CROSSTALK does not pay for submissions. Articles published in CROSSTALK remain the property of the authors and may be submitted to other publications.

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

**Trademarks and Endorsements:** This DoD journal is an authorized publication for members of the Department of Defense. Contents of CROSSTALK are not necessarily the official views of, or endorsed by, the government, the Department of Defense, or the Software Technology Support Center. All product names referenced in this issue are trademarks of their companies.

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

**STSC Online Services:** www.stsc.hill.af.mil  
Call (801) 777-7026, e-mail: stsc.webmaster@hill.af.mil

**Back Issues Available:** The STSC sometimes has extra copies of back issues of CROSSTALK available free of charge.

**The Software Technology Support Center** was established at Ogden Air Logistics Center (AFMC) by Headquarters U.S. Air Force to help Air Force software organizations identify, evaluate, and adopt technologies to improve the quality of their software products, efficiency in producing them, and their ability to accurately predict the cost and schedule of their delivery.



## 2003 U.S. Government's Top 5 Quality Software Projects



For years, the department's acquisition community has recognized software as a key parameter for enabling the performance of our defense systems. That is an understatement to say the least because of the significant role that software plays in making it possible to undertake the acquisition of complex systems to meet our dynamic defense needs. Software helps us address the complex command, control, and communication issues relating to a network-centric battlespace; increase the effectiveness of our information and intelligence systems; improve the efficiency of our logistics systems; and provide the joint, multi-mission capacity required to meet evolving warfighter capability needs.

The award-winning projects highlighted in this issue of *CROSSTALK* demonstrate the continued ability of industry to meet the department's acquisition needs. They are a testimony to our combined ability to successfully conceive, design, field, and sustain complex defense systems. I congratulate the program managers and their government, industry, and academia teams for their achievement of this award.

Our strong reliance on software as a critical performance driver has significant implications within the acquisition community. More often than not, poor system performance is blamed on software; but at times, software is called upon to compensate for system performance issues attributable to non-software components. In either case, it is clear that software bears a significant responsibility for a system's ability to perform its intended mission, which is why the department is emphasizing the acquisition of software in a systems engineering context.

Within the Department of Defense, a major focus on improving defense systems acquisition is derived from the Honorable Michael W. Wynne (acting under secretary of defense for Acquisition, Technology and Logistics) who said that it is imperative to "... help drive good systems engineering practice back into the way we do business." In the context of software, this sends a clear message to the community that we must continue to embrace software engineering as a critical and integral part of a comprehensive, capabilities-based systems engineering approach to acquisition.

Again, I congratulate the program managers and their industry partners, and encourage them to share their lessons learned and best practices to further promulgate the successes of these five projects across the greater acquisition community.

David R. Castellano  
*Deputy Director, Systems Engineering*  
*Defense Systems*  
*Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics*

# Winning Projects Exemplify Success for Developers and Acquirers

Elizabeth Starrett  
CROSSTALK

*The Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics and the CROSSTALK staff announce the 2003 U.S. Government's Top 5 Quality Software Projects. Again, this year's winners represent great success in delivering software to government acquirers.*

## U.S. Government's Award-Winning Quality Software Projects

The results are in, and it is clear that the government is building many successful software packages that are top-notch examples in project management and quality control. Listed below in alphabetical order are the winners of the third annual U.S. Government's Top 5 Quality Software Projects contest managed by CROSSTALK. We congratulate them and hope you enjoy reading more about their winning projects in the following pages.

- **Advanced Field Artillery Tactical Data System**  
Customer: U.S. Army/USMC/  
U.S. Navy  
Developer: PM Intelligence and Effects and Raytheon Team
- **Defense Medical Logistics Standard Support**  
Customer: Military Health System  
Developer: DMLSS Program Office
- **H1E System Configuration Set**  
Customer: Program Manager Air, AIR-265  
Developer: F/A-18 Advanced Weapons Laboratory and Boeing IDS
- **OneSAF Objective System**  
Customer: Program Manager OneSAF Objective System – U.S. Army's Program Executive Office for Simulation, Training, and Instrumentation  
Developer: OneSAF Objective System Integrated Product Team
- **Patriot Excalibur**  
Customer: AFMC  
Developer: 46 TW/XPI (TYBRIN)

The Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics and the CROSSTALK staff announce the 2003 U.S. Government's Top 5 Quality Software Projects. This year's winning projects continually impressed reviewers and judges through four rounds of scoring.

While reflecting on this year's winners, I found it interesting that over 40 percent of the scoring criteria focuses on how happy the customer is with the end product, and less than 20 percent focuses on the processes used to develop the software. However, again this year, the winning projects implement software development processes that involve peer reviews, configuration management, requirements management, and other practices suggested by ISO 9001 and various software maturity models.

Based on this, I decided to look for additional consistencies among the projects. Another interesting commonality I found in reviewing project information is the requirement for these projects to interface their software with other software. This was also a common theme in many of CROSSTALK's January 2004 articles from senior military leadership: the requirement for software to be able to network together for information requirements. Many of this year's Top 5 winners do just that.

Customer support always impresses me. In the Advanced Field Artillery Tactical Data System (AFATDS), the Raytheon engineers provided support in the conflict zones in Iraq where the AFATDS was being used. The Patriot Excalibur developers provide in-house training at no charge and if the users want training at their location, they only need to fund the cost of the temporary duty.

Physically locating the developers with the customers and users during development also seemed to be a big help in the success of some of the winning projects.

Given the rate of software project failures and the current U.S. government focus on acquisition process improvement, I asked customers of these projects to provide

<sup>®</sup> The Capability Maturity Model and CMM are registered in the U.S. Patent and Trademark office by Carnegie Mellon University.

some tips for acquisition success. Some pointers I received include the following:

- Institute an Alpha Contracting (AC) process, which places a heavy focus on requirements definition. All stakeholders (users representatives included) participate, ensuring adequate detail to the requirements to preclude misinterpretation during software development. This is done prior to contract award and also allows a more accurate sizing of the effort. AC also fosters an open communication that carries on through the development. The AC process jump-starts the team.
- Consider using multiple development contractors to minimize the risk (impact of failure) if one of the developers fails.
- If possible, have the contractors physically collocated with the government team enabling daily communication.
- Consider facilitating collaboration with a Web-based collaborative development environment.
- Consider the spiral development methodology and an implementation of eXtreme Programming.
- Consider heavy use and participation in open source software.
- Utilize advanced software development processes such as the Capability Maturity Models<sup>®</sup>.
- Select a proven government/industry team with experience with the intended end-user operating environment.
- Develop detailed plans and agreed upon methods and metrics to measure progress, then work the plan. Whenever progress deviates from the plan, assemble the team to establish corrective actions, then track those actions to closure. It can be very rigorous and sometimes tedious but the result is worth it.

I would like to give special thanks to our final judges whose respect in the software community adds prestige to this award. I would also like to thank Mark Schaeffer, principal deputy, Defense Systems, Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics for continued sponsorship of this award. ♦

## TOP 5 QUALITY SOFTWARE PROJECTS JUDGES' BIOGRAPHIES



**David A. Cook, Ph.D.**, is a senior research scientist at The AEGIS Technologies Group, Inc., working as a verification, validation, and accreditation agent in the modeling and simulations area. He is currently supporting the Airborne Laser program and has more than 30 years experience in software development and management. He was formerly an associate professor of Computer Science at the U.S. Air Force Academy, a deputy department head of the Software Professional Development Program at the Air Force Institute of Technology, and a consultant at the U.S. Air Force Software Technology Support Center. Cook has published numerous articles on software-related topics. He has a doctorate in computer science from Texas A&M University. <dcook@aegistg.com>



**Carol A. Dekkers** is president of Quality Plus Technologies, Inc., a management consulting firm specializing in creating peace of mind for companies who want to improve their software processes. Software measurement, software quality, process improvement, requirements, and software sizing (using function point analysis, as an example) are a few of the Quality Plus areas of specialization. Dekkers is also the chair of the American Testing Board, the U.S. participant in the International Software Testing Qualifications Board Certified Software Tester certification. She is a Certified Management Consultant, a Certified Function Point Specialist, and a professional engineer (Canada). She holds positions with the Project Management Institute, the American Society for Quality (ASQ), and the International Organization for Standardization's software engineering standards subcommittee. ASQ's Quality Progress named her one of 21 New Voices of Quality for the 21st Century. <dekkers@qualityplustech.com>



**Jack Ferguson, Ph.D.**, is manager of the Appraisal Program at the Software Engineering Institute where he is responsible for the training, authorization, and quality of appraisers for the Capability Maturity Model<sup>®</sup> for Software (SW-CMM<sup>®</sup>) and CMM Integration<sup>SM</sup> (CMMI<sup>SM</sup>) models who use the Capability Based Assessment for Process Improvement and the Standard Capability Appraisal Method for Process Improvement methods. Previously, Ferguson was director of Software Intensive Systems in the Office of the Secretary of Defense. Before that, he led the teams that developed the Software Acquisition CMM and the CMMI models. Ferguson has a doctorate in aerospace engineering and is listed in Jane's

Who's Who in Aerospace for his work on Global Positioning System spacecraft control systems. <jrf@sei.cmu.edu>



**Lt. Col. Ricky Sward, Ph.D.**, U.S. Air Force, is an associate professor of computer science at the U.S. Air Force Academy. He is currently the deputy head for the Department of Computer Science and the course director for the senior-level two-semester Software Engineering capstone course. Sward received his doctorate in computer engineering at the Air Force Institute of Technology in 1997 where he studied program slicing and reengineering of legacy code. <ricky.sward@usafa.af.mil>



**Edward C. Thomas**, director, U.S. Army Communications-Electronics Command leads efforts to provide state-of-the-art software engineering products and services throughout the U.S. Army and the Department of Defense. These products and services include enterprise-level software architecting and integration; software technology assessment and application; system-level software engineering for more than 400 individual Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance and business systems; and worldwide technical support to war fighting units. He leads a global organization of more than 3,000 military, civilian, and industry employees and manages an annual budget of approximately \$300 million. Thomas has worked in various capacities for the Army since 1974 and was appointed to the Senior Executive Service in 2001. <edward.c.thomas@us.army.mil>



**Richard Turner, D. Sc.**, is a research professor in engineering management and systems engineering at The George Washington University. In support of the U.S. Department of Defense, he supports the Systems Engineering Directorate of the Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics, Defense Systems organization in assessing software aspects of weapon systems programs, implementing software acquisition process improvement programs, and identifying and transitioning new software technology into defense systems. Turner is a co-author with Barry Boehm of "Balancing Agility and Discipline: A Guide for the Perplexed," and co-author with Dennis Ahern and Aaron Clouse of "CMMI<sup>®</sup> Distilled." He has a Bachelor of Arts in mathematics from Huntingdon College, a Master of Science in computer science from the University of Southwestern Louisiana, and a Doctor of Science in engineering management from The George Washington University. <rich.turner.ctr@osd.mil>

<sup>SM</sup> CMM Integration is a service mark of Carnegie Mellon University.  
<sup>®</sup> CMMI is registered in the U.S. Patent and Trademark Office.

# The Advanced Field Artillery Tactical Data System Proves Successful in Battle

Pamela Palmer  
CROSSTALK

*While Operation Iraqi Freedom remains ongoing, one initial military success is the operation of the Advanced Field Artillery Tactical Data System (AFATDS) in preventing friendly fire accidents in missions covering hundreds of miles of territory and hundreds of maneuvering combat units. With the AFATDS', robust communication architecture, the entire theatre was provided with a common understanding of the battlefield fire support situation.*

One of the U.S. government's software success stories for 2003 is the Advanced Field Artillery Tactical Data System (AFATDS). In Operation Iraqi Freedom (OIF), the AFATDS prevented friendly fire accidents, provided additional protection to friendly forces, and created significant savings in weapon systems and ammunition costs. All this was accomplished through AFATDS' timely and effective fires delivered against enemy targets in accordance with the commander's guidance.

In addition, as specifically noted by Lt. Gen. Steven W. Boutelle, the U.S. Army's chief information officer, the AFATDS has been a model procurement program – on schedule, within budget, and meeting all technical performance standards and contractual delivery requirements.

The AFATDS' greatest contribution in 2003, however, was its outstanding performance by the 600 systems used by the front-line fire support units of the U.S. Army and Marine Corps during OIF. Maj. Gen. Michael Maples, former commanding general of the U.S. Army Field Artillery Training Center, noted in July 2003 that the AFATDS was a significant combat multiplier that helped military units win the war, and prevented friendly fire incidents from occurring during OIF<sup>1</sup>.

Overall, the AFATDS has become an integral part of the Army and Marine Corps command and control (C2) network-centric architecture – delivering devastating and accurate fires at the right place and right time. Additionally the system is proving to be a key transformational agent as the Army moves to the future force designs from its current force. While OIF remains ongoing, the AFATDS continues in its role of preventing friendly fire incidents, shortening times for effective engagement of targets, and creating significant savings in hardware and ammunition costs.

## The Inner Workings

The AFATDS is a multi-service U.S. Army/Marine Corps automated C2 system for fire support used throughout the battlefield at all levels. The AFATDS provides timely and effective fires delivered against enemy targets in accordance with the commander's guidance. Built from the bottom up, the AFATDS processes, analyzes, and exchanges combat information within the fire support architecture and the joint environment. "It knows where every fire support platform is located on the battlefield, the ammunition status, the range capability, etc.," said Lt. Col. James J. Chapman, product manager Fire Support Command and Control, Ft. Monmouth, N.J. "AFATDS uses a robust communication architecture that provides the entire theatre with a common understanding of the fire support battlefield situation," he explained.

The AFATDS includes interoperability with other Army battle command systems, coalition systems, Marine Corps C2, intelligence and sensor systems, the Air Force's Theater Battle Management Core System, and the Naval Fires Control System. The AFATDS is capable of managing and tasking weapon systems from the joint community, including field artillery cannons and rockets/missiles, fixed wing air fire support, Naval surface fire support, mortars, and Army/Marine aviation (helicopter) attack systems. The AFATDS performs these functions at echelons from above corps down to platoon level.

On the battlefield, the AFATDS provides operators with a complete look at all the engagement target options available to attack a target. "A sensor, such as a human or radar, detects an enemy target to engage as a threat to be eliminated," said Steve Bohan, technical lead at Raytheon. "The sensor submits a digital or voice request to AFATDS, reporting the type of target, location, and engagement instructions. AFATDS compares the target data to com-

mander's orders and available weapon platforms, and develops options for all these different platforms to destroy the target. All options, along with AFATDS' recommendations are presented to the operator for review. The operator can configure AFATDS to make recommendations considering what is important to him: speed of engagement, time, munition, preferences, etc. After the target is engaged, AFATDS tells the operator when the fire is completed, whether to fire more, and the effects reported by the sensor; it then distributes this information across the battlefield."

The AFATDS software provides functionality in four major areas: situational awareness, battle planning, battle management (execution), and fires/effects processing. It provides target analysis and weapon selection logic that ensures that the right munitions are placed on the right target at the right time.

## Several Quality Aspects

Quality is critical to the AFATDS since it is being used in war as well as in live-fire exercises where personnel safety is essential. Key to achieving quality is the use of the Capability Maturity Model<sup>®</sup>, processes, Raytheon Six Sigma, and a comprehensive software cost estimation model. The AFATDS System was developed at Raytheon's facilities in Fort Wayne, Ind. It uses an incremental build approach with a series of sequential releases containing increasing functionality. Each build goes through a full development life cycle, including software requirements analysis, design, code, unit test, and integration.

The AFATDS has been certified to meet safety, security, and Defense Information Infrastructure Common Operating Environment Level 6 requirements. It interfaces with more than 50 different digital systems across the joint spectrum and is required to run on an ever-changing series of hardware platforms. The AFATDS has more than 5,000 system

requirements and over 8,500 software requirements. These requirements are documented in the System Software Specification, Interface Control Document, and 16 Software Requirements Specifications that total over 9,000 pages.

An extensive configuration control system using Apex and Encompass tools for the software environment assisted software engineers in producing quality code. The DOORS database is used for tracking system and software requirements, and produces bi-directional traceability between these documents and test cases. Engineering change requests are used to control changes to the system requirements, and change orders provide the authorization methods for working controlled software changes.

“Software development tools allow us to have top-to-bottom traceability all the way back to system-level requirements,” said Bohan. “Since DOORS is object oriented, we can run queries and get reports on changes, including printouts. It’s a good tool for online requirements documents and can search through the 9,000 pages of requirements developed.”

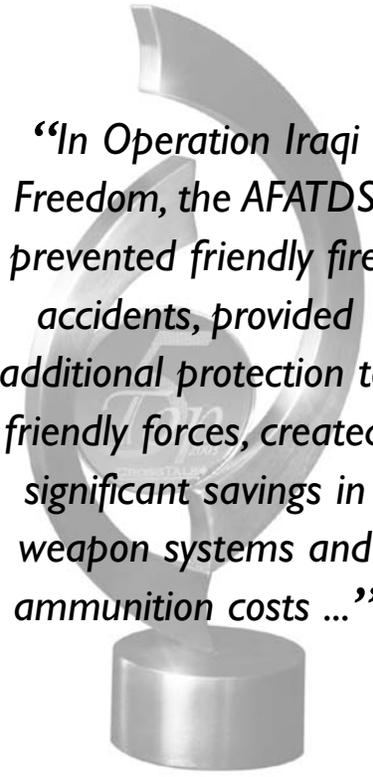
Quality measurements are continuously performed and documented in weekly and monthly metrics reports. These reports are generated to track defect density, number of requirements integrated to date, fault correction progress, software changes initiated versus retested successfully, and tasks started measured against tasks completed to date.

“Early versions of AFATDS used traditional waterfall development,” said Cynthia Inteso, technical lead at Ft. Monmouth. “As time progressed, each AFATDS version became more complex and since we had an established software product baseline, we transitioned to an incremental iterative build approach.” This meant developing a series of builds, each with a small set of requirements. When rolled up at the version level, these incremental builds allowed a more rapid approach of providing quality-enhanced capability to the user in the field.

Another facet of the quality process uses a series of test procedures termed a smoke test. A smoke test is run on each build that exercises the main threads of the system assuring build integrity and software quality early in the development cycle. After code completion, a full integration test suite is run over a 10-week period to assure defect-free software at final delivery. During recent government testing, there were zero priority one or two software defects reported in the 1.8 mil-

lion lines of code.

Testing was a big reason that a small number of problems were reported from locations in Kuwait and Iraq. “Our processes enforce traceability of requirements through modules and into test cases,” said Bohan. The government also runs independent verification and validation, providing a separate set of eyes to review software per criteria, he added. “Testing will often show that a system meets requirements, but then you see additional requirements to add. Everything in this system is configuration managed for complete control and repeatability.”



***“In Operation Iraqi Freedom, the AFATDS prevented friendly fire accidents, provided additional protection to friendly forces, created significant savings in weapon systems and ammunition costs ...”***

Communication was the main contributor to the AFATDS’ quality ratings, according to Chapman. “Quality really links back to open communication between team members, so we were all clear with regard to where we were going.”

### **Combat Success**

The AFATDS V6.3.1 software was materiel released to various Army and Marine units in January 2003 and was immediately deployed for use in OIF. This was the first time the AFATDS was used in combat. Raytheon engineers provided support in the conflict zones. Combat-inspired enhancements and problems were immediately reported back to Raytheon through various methods via the Raytheon Field Integration Team.

The field engineers provided e-mail problem definitions and information that

enabled the Fort Wayne engineers to create and debug the problem in a lab environment. Semi-weekly teleconferences with the engineers also helped accomplish problem resolution in a timely manner. The outstanding performance of this software was demonstrated by the small number of problems reported from locations in Kuwait and Iraq.

According to one customer, the return on investment from the AFATDS in performing its stated mission is best measured by the results of OIF. While directing the fires of more than 35,000 rounds of munitions, 857 rockets, and 453 long-range missiles, the AFATDS prevented friendly fire accidents (fratricide) from occurring among its users. The AFATDS’ coordination of air space alone allowed friendly fixed- and rotary-wing aircraft to safely and simultaneously engage enemy targets along with friendly rockets, missiles, and land and naval gunfire without the loss of an aircraft due to friendly fire.

The AFATDS has proven itself against the test of time. In the hands of warfighting units since 1997, the program has accommodated its end-users (Army, Marine Corps, and Navy) by incorporating the most current operational techniques, as well as modern technologies. Additionally, the AFATDS program has been a role model for achieving the critical capabilities development, as well as the contractual necessities of maintaining cost, schedule, and performance. The AFATDS’ future appears to be as bright and opportunistic as its past, since the AFATDS will clearly be part of the transition to the future fires C2 systems for joint and coalition operations. ♦

### **Note**

1. AFATDS Media Day, July 2003, Rosslyn, VA.

### **Project Points of Contact**

**Cynthia Inteso**  
**PM Intelligence and Effects**  
**Technical Lead – AFATDS System**  
**Development**  
**Phone: (732) 532-6004**  
**E-mail: cinteso@c3smail.**  
**monmouth.army.mil**

**Lt. Col. Jim Chapman**  
**PM Intelligence and Effects**  
**Product Manager, Fire Support C2**  
**Phone: (732) 427-3328**  
**E-mail: james.chapman@c3smail.**  
**monmouth.army.mil**

## The DMLSS Program Brings Electronic Commerce to the Military Medical Treatment Facilities

Pamela Palmer  
CROSSTALK

*The Defense Medical Logistics Standard Support product suite establishes an electronic commerce solution that has transformed the acquisition and distribution of medical supplies to both peacetime hospitals and deploying forces. A just-in-time inventory system provides medical supplies within hours, and lower priced regional contracts with dedicated vendors have reduced costs.*

The Defense Medical Logistics Standard Support (DMLSS) product suite has given a whole new meaning to the term *modern medicine*, or more specifically to the *delivery* of modern medicine. DMLSS establishes an electronic commerce solution that has radically transformed the acquisition and distribution of medical supplies to both peacetime hospitals and deploying forces. Gone are the massive amounts of medical inventory in depots and retail activities. Instead, a modern just-in-time inventory system provides medical supply support with response times measured in hours, not days or weeks.

"To support the hospitals in the 1990s, medical logisticians in all three services used manual processes to take requirements, including filling out paper requisitions to give to the depot [warehouse distribution center]," said Col. Cathy Erickson, the DMLSS program manager, Joint Medical Information Systems Office, TRICARE Management Activity, Falls Church, Va. "Stored at the depots were six to 12 months of medical supplies. Turnaround time was 30 to 45 days. Every receiving document was done manually, including processing through the financial system. Payment to commercial vendors could take six months. Manual methods were also used for inventory and equipment records."

With DMLSS, medical supplies are not only delivered faster and more reliably, they are also substantially less expensive. High-priced local contracts between medical treatment facilities and local pharmaceutical companies have been replaced by high-volume, lower priced regional contracts with dedicated vendors through the Prime Vendor program, Erickson explained. These savings will only increase annually due to automated tools and a comprehensive database integrated with commercial purchasing and electronic

commerce practices. "This change in the acquisition process has been pivotal to transforming the military medical logistics supply chain into an industry model."

Using Department of Defense (DoD) communications capabilities, forward medical units are now able to order materiel directly from commercial distributors using standard electronic commerce transactions in a fraction of the time, at a fraction of the cost. The DMLSS' Web-based ordering initiative allows customers to electronically browse, compare, select, and order non-prime vendor items via the Internet, while retaining the ability to electronically interface with existing medical and financial systems.

"A drug is ordered electronically. The order is transmitted electronically to the Prime Vendor who fills the order the next day. Payment is done electronically and received within 30 days," said Erickson. "This reduced facility stock has allowed us to reduce depot supply to military unique items."

The full range of the DMLSS product suite also provides capability for inventory management, facility management, equipment and technology management, Web-based customer support, business intelligence, customer area inventory management, and assemblage management. The Prime Vendor program provides access to more than 1 million medical materiel items.

"We now have electronic cataloging, electronic money management, and quality assurance checking embedded for Federal Drug Administration recall information on drugs or equipment," said Erickson. "We also receive pricing updates as recent as 30 days from negotiation versus six months, reducing risk of hospitals' ordering at a wrong price. In all, for every \$1 spent on the DMLSS, \$6.40 is returned to facilities to deliver healthcare. And we're not done yet."

### Years in the Making

The DMLSS automated information system was developed in three releases. Release 1, deployed in the 1996-1997 calendar years, included a facility management and product and price comparison capability. Release 2, deployed in the 2000-2001 calendar years, added customer area inventory management capability that included hand-held terminal technology. Release 2 also included a Web-based customer support module and an improved Prime Vendor interface.

Release 3, currently being deployed worldwide to 110 medical treatment facilities and to 86 very small sites, adds equipment management and maintenance, higher level inventory management, and readiness assembly management capabilities. Release 3 replaces service legacy systems in DoD medical treatment facilities. It also provides a comprehensive readiness management program allowing a seamless, automated, and efficient transition from peace to contingency/war. It operates on varied platforms in support of deployed operations and has the flexibility required to meet military service requirements.

The DMLSS system is a complex, client-server system containing approximately 3.5 million lines of software code. This code includes customized medical logistics functionality developed at the program's development facility, as well as five client-side and 17 server-side commercial off-the-shelf products that have been fully integrated into the system design, explained Debbie Bonner, Director of Operations at the DMLSS Program Office. "All military groups met in a joint development group to turn in specific requirements. A configuration control board put the requirements together and sent them to the services medical logistics chiefs for approval. Those requirements were then put into

very specific SHALL statements.”

Three types of inputs were used to develop requirements, said Lt. Col. Ken Darling, Director of DMLSS Business Process Reengineering. “The user provided direct input in changes to the system. The medical logistics chiefs each provided direct input needs independent of enterprise needs. Third was to consider what is available in the industry today.” This last consideration was made to build language code into the system to support technology that is years away in development such as embedded tags on pallets that can be machine read.

To make requirements changes requires a meeting with users from all three services, explained Bonner. “Requirements are defined using standard terminology that is built into our system. Users meet to hash out new requirements and develop concurrent terminology that becomes part of that standard language. They can use the system differently, but we maintain standard terminology.” The DMLSS development team remains very close to its users. A representative from each service sits in the program office, two representatives are in the development center, and a third splits his time between Washington, D.C. and the development center. “Lots of communication is a big part of our process,” said Bonner.

More than 200 development workstations of up-to-date configurations, with several operating systems simulate the various field environments. The database engine is currently Informix, however, a new effort is underway to convert to an Oracle database engine. “The conversion is necessary because the depots support the battlefield theatre, and they are ahead of us using Oracle. This will put us on one platform in peacetime and wartime.”

“Another advantage is cost savings,” said Darling. “Our current platform costs about \$25,000 to \$30,000. We think the Oracle platform will cost less than \$5,000. There will also be cost advantages in maintenance and overhead.”

The DMLSS team uses a Rapid Application Development approach to software development and sustainment, and has refined the configuration management and functional testing process to the extent that it can support rapid evolution and quality assurance goals, as well as rapid deployment to the field for all software upgrades and corrective actions. Functional and technical experts test each release of the system at the development center. Defects are corrected through quality assurance steps in the process. Each development contractor is required

to provide CMM Level 3-like processes.

Peer reviews and code walkthroughs are used for early problem identification. Developers conduct unit testing as a part of each software build. Integration testing is a regular process in the effort to produce a DMLSS release. Special testing plans are put in place for designated system-wide business processes. Periodically data anomalies and user-driven problems occur in fielded systems. The DMLSS development center addresses these anomalies through a modification of the data conversion process. The configuration management organization is able to



**“ ... forward medical units are now able to order materiel directly from commercial distributors using standard electronic commerce transactions in a fraction of the time, at a fraction of the cost.”**

rapidly patch data through a defined and repeatable process and keep sites running with uninterrupted operations.

Each development contractor is required to report financial and schedule performance according to an earned value methodology. As such, cost performance and schedule performance curves are calculated and delivered as a part of a contractually required monthly progress report. These reports are reviewed with DMLSS program and development center leadership on a monthly basis during contractor financial reviews. The latest cumulative indices show a cost performance index of 1.06 and a schedule performance index of 0.98.

### Lessons Learned

In looking back Bonner said, “Nothing succeeds like success. Each time we put

out a release, we learned from our mistakes.” There was a lot of learning with software and hardware deployment. Bonner said she realized how important it was to have weekly meetings within the services to determine when they were coming for installation and the amount of training room space needed. “The first release was Russian roulette to deploy, but much more communication with later releases made it a smooth delivery. We learned it was very important that every entity on the deployment team needed to be on the same page through weekly conference calls.”

Erickson agreed, adding, “We would show up and the chief information officer would say that he didn’t know we were coming, so we couldn’t tap into their network.” Erickson said they learned it was important to know the hospital needs ahead of time and have the users develop the deployment schedule. “Then we held them on task to stay on schedule, replacing deployment facilities if one backed out.”

The DoD medical user community has embraced DMLSS’ changes in acquisition strategy and its pioneering electronic commerce efforts. The Prime Vendor initiative capitalizes on the efficiencies of commercial distribution channels, reduces procurement times from up to 45 days to two days or less, reduces inventory by up to 85 percent, and has a 95 percent fill rate in less than 24 hours. Web-based ordering provides customers a wide variety of commercial items at negotiated and very competitive prices and expedites payment through the acceptance of standard military electronic transactions and government-issued credit cards.

The new DMLSS emphasis is on smarter, more cost-effective acquisition through electronic commerce, committed volume purchasing, and long-term partnership with suppliers. The DoD medical user community now has at their fingertips a fully automated and integrated inventory and information management system coupled with the best acquisition business practices in place enabling them to meet the medical logistics needs of the military services in the 21st century. ♦

### Project Point of Contact

**Col. Cathy Erickson**  
**DMLSS Program Manager**  
**Phone: (703) 575-9771**  
**E-mail: cathy.erickson@tma.osd.mil**

# The H1E System Configuration Set Lays the Foundation for Decades to Come

Pamela Palmer  
CROSSTALK

*Recoding 1.3 million lines of F/A-18 assembly language code to a more cost effective High Order Language (HOL) has made every piece of aviator functionality fast, modular, and inexpensive enough to ensure that aircraft capabilities can be expanded for years to come. The HOL is a significant leap forward in flexibility of computer code and test efficiency.*

While it took 20 years to create the functionality in the Navy's current fleet of 950 F/A-18 aircraft, the Naval Air Systems Command (NAVAIR) government-industry team recently fielded the High Order Language Version 1 F/A-18E and F (H1E) System Configuration Set (SCS) that recoded 1.3 million lines of F/A-18 assembly language code to a more cost-effective High Order Language (HOL) in just five years. In addition, every warfighting function was verified in two years of intensive lab and flight-testing. Simultaneously, new hardware for the mission computers and displays was created, and is considered part of the SCS. The HOL is supporting aircraft production schedules of 400 F/A-18 E/F aircraft.

The project's goals were ambitious. Make every piece of aviator functionality fast, modular, and inexpensive enough to ensure that aircraft capabilities can be expanded for years to come. The challenge was to create new hardware and software to work in a real-time combat environment while meeting production line schedules. The risk involved simultaneously changing hardware and software to the U.S. Navy's primary aircraft. Finally, not letting costs escalate was a key requirement.

## A Multitude of Innovations

The idea to convert the F/A-18's real-time processing from assembly language to the more efficient HOL originated with the manufacturer, Boeing Integrated Defense Systems. "Boeing recognized the direction we were heading, so they put their independent research and development (IRAD) dollars into getting it started," said Harlan Kooima, H1E project manager. "Then we took over the idea for completion."

The project's requirements were based on the detailed documents developed over 20 years that describe how the F/A-18 operates. "The basic requirement was to make the new software package look and function the same as the previously fielded system," said Kooima. "Any deviations were captured and stated in written docu-

ments. "A solid understanding of requirements was key to our success."

The software was redesigned from a top-down approach to an object-oriented design. In the legacy system, written in CMS2 assembly language, rehoused functions were recoded in C++. A significant investment was made to ensure the architecture supported on-demand, all-the-time requirements of a real-time system, while being modular and easily maintainable. "Today we have much better structured software that has good partitioning," said Kooima. "When we make changes in one area, it does not induce problems in another area. For example, if a change is made to a radar module, we have a high level of confidence that it won't affect the radios."

"There are also benefits transitioning to a layered software architecture," said Marty Montgomery, H1E software manager at Boeing Integrated Defense Systems. "In testing, we were able to adapt quickly to multiple versions of target hardware and low-level software with only a few problems."

This \$160-million software and \$210-million hardware project involved more than 100 major warfighting capabilities such as Heads-Up Display and Backup Mode, with more than 1,000 possible operator selections. According to Kooima, there were just 166 instances of differences in operator interfaces between the legacy system and the HOL conversion. These were understood and negotiated prior to implementation.

"Our goal was to be as close to the fielded legacy system as we could be," Kooima said. "Out of the million-plus features the operator uses on a mission, we kept the same basic commands he is used to. We didn't change his life."

Commercial off-the-shelf (COTS) products were leveraged to automate code generation. The development environment consisted of real-time models running on Silicon Graphics workstations and a debugger tool set running on a Sun server. The project team created a new capability mak-

ing the entire mission computer Operational Flight Program (OFP) available on a user's desktop computer for user interface development, training, and debugging. The desktop environment (DTE) allowed developer tests to occur on a workstation versus a separate test facility. The DTE mitigated risks associated with parallel hardware/software development and was acquired for use on AV-8B. Also the innovation of an automatic display code generator shows promising use in flight simulations, test facilities, trainers, and technical publications.

"The portability of the commercially based flight software and its layered architecture makes it usable in simulators and in trainers," said Montgomery. "COTS tools have allowed us to prototype and advance our final display software, and that has helped reduce cycle time and errors. The DTE has the same type of capability in non-display software and has really impacted the quality of what we take to the target."

Kooima added that the COTS-based system is the enabler for future capability enhancements. "It allows us to grow and add more computing horsepower on demand, for example, to expand the F/A-18's use into an electronic attack role. We've made updating the aircraft's entire functionality more modular, economical, and faster."

The H1E SCS hardware was built from the ground up. The F/A-18 E/F Advanced Mission Computer (AMC) is a totally new development and a move to commercial-based architecture for the hardware, said Montgomery. The two AMCs contain six processor modules each, and are connected using a high-speed fiber channel. There were unique challenges for COTS, he said. From a software standpoint, the biggest challenge was for these intense, embedded software applications to have the built-in software test capability to perform debugging. COTS products have fewer capabilities than our custom hardware developments. So the DTE was built for this reason. "Due to the layered architecture, we

could run the OFP on our desktops and use Microsoft Studio to mature the product before we went to the target hardware. This minimized the number of undetected bugs.”

To deal with supplier changes to COTS products, Kooima said, “We have a set configuration baseline. The hardware supplier can make changes to the baseline as long as the functional equivalent is still there. For example, an integrated circuit can change as long as the supplier ensures it is the functional equivalent when it is done.

In another major hardware enhancement, processing for the displays was put inside the computer versus inside the display head as it was on the legacy system. The Engenuity Technologies, Inc. Virtual Application tool makes cockpit display generation more like desktop displays and is based on commercial standard, OpenGL. As a result, this hardware allowed the team to use commercial tools to write more than 40 percent of the software at a much-enhanced productivity rate. It saved a lot of time and money.

### Quality and Test Measures

In testament to the quality in the project, Boeing matured from a Software Engineering Institute Capability Maturity Model® (CMM®) Level 2 to a Level 5 using H1E SCS as part of its assessment. While there were trade offs in reaching Level 5 while completing the project, Kooima said he still believes doing it was a benefit. “Since we were developing totally new software from scratch along with new hardware, we didn’t have to make changes to baseline processes and tools with the CMM.”

The H1E SCS demanded more coordination than previous programs, said Kooima. It involved two program executive officers, two different N-78 sponsors, a major aircraft delivery program, and two fleet squadrons. Each delivery consisted of up to 1.4 gigabytes of data and executables. The test effort was huge in scope. “We weren’t focused on the deltas from a previous baseline. We had to look at the entire F/A-18 system with fresh eyes and efficiently test everything from the bottom up.” The total integration test effort for the H1E SCS was 3,000 hours and 500 flights.

Testing really was a build up approach, said Kooima. The lowest testing level was done on software engineers’ desktops. From there it migrated to the software test facility that would run the software on real mission computer hardware. Then it went to the F/A-18 Advanced Weapons Laboratory (AWL) where it underwent full system integration testing on real, integrat-

ed avionics systems, aircraft ground- and flight-testing. “The entire focus on finding and resolving errors early was extremely successful.”

While it is still controversial, Montgomery said that a real concerted effort was made to get into functional capability testing as quickly as possible; the DTE made this possible. “We did not do low level unit tests; instead we went straight into functional desktop testing. We saw the benefits.”

Boeing identified three quantitative goals to ensure software quality: cost of less than 1.5 labor hours per line of code, delivery of less than 0.5 defects per thousand lines of code, and maintaining cost performance indices and schedule performance indices of greater than 0.95.



Montgomery said these goals were reviewed weekly and were successfully met throughout development.

The team relied on the AWL’s proven processes for measuring system maturity before determining a product is mature and ready for operational testing. Kooima said that other valuable F/A-18 processes were its risk identification and mitigation processes, as well as a rigorous process for assessing the risks and costs associated with changing requirements. The AWL’s process for managing changing requirements calls for agreement by a NAVAIR Level 1 lead before the change is accepted into the project.

Kooima said that the H1E SCS’s

comprehensive metrics approach provided insight into product, project, and process quality throughout development. A summary of H1E plans versus actual metrics follows and has been independently verified.

- **Requirements Control.** Full functionality to the initial plan was delivered. Requirements scope was expanded to provide additional functionality.
- **Source Lines of Code.** This effort was primarily a conversion of 1.3 million lines of assembly code to HOL; however, it also included 3.8 percent growth in new, sponsor-requested warfighting capabilities and systems.
- **Schedule.** Product delivery occurred in the month promised.
- **Software Engineering Productivity.** Productivity for software engineering of legacy F/A-18 systems is 3.5 hours per line of code. The H1E SCS achieved a rate of 1.27 hours per line of hand-generated code.
- **Defect Density.** The number of escaped defects is .007 per thousand lines of source code for the first 90 days of operational use.
- **Test Activities.** Test activities (hours, type, anomalies) were tracked to ensure complete coverage of requirements.
- **Staffing.** Began the program with a staff of 40 C++ programmers. At its peak, which corresponded with the dot-com demand for experienced programmers, the H1E SCS utilized 180 software developers.

### The Benefits Continue

In an added reuse bonus, the AV-8B program is picking up the H1E SCS software processes for use during the later phases of its own HOL conversion. Both programs create similar types of weapons and sub-systems, said Kooima. They were going through a similar upgrade program and Boeing was the prime contractor. Boeing again recognized the business opportunity and moved forward with the reuse using IRAD money.

The H1E SCS lays the foundation for the E/F aircraft of the future, said Kooima. Using the HOL language is a significant leap forward in flexibility of computer code and test efficiency. It is the foundation on which new big-ticket, acquisition systems like Active Electronic Scanned Array can be built and fielded in less time. It provides growth for expanding the aircraft’s utilization to support the Navy’s need for a replacement to the EA-6B. Indeed, the H1E SCS lays the foundation for the Navy’s aircraft advancements for decades to come.◆

# The OneSAF Objective System Fits Individual Simulation Needs

Chelene Fortier-Lozancich  
CROSSTALK

*Can battlefield simulation software be everything to everyone? The challenges faced by the One Semi-automated Forces (OneSAF) Objective System found out the answer when faced with the Army's modeling and simulation needs, and in the process set a new standard for what they did and how they did it.*

Computer modeling software must meet many requirements: there must be common pieces, the components must be flexible for different requirements and be able to meet a user's particular need, and the software needs to be *everything to everyone*.

This challenge was faced by the U.S. Army's modeling and simulation division: how to address a broad range of requirements for a flexible simulation battlefield modeling architecture with a supporting set of components, tools, and services that allows individual users to compose a simulation to meet their individual needs.

This is where the One Semi-automated Forces (OneSAF) Objective System (OOS) comes in. The OOS is composable, next-generation Computer Generated Force (CGF) modeling software that represents a full range of operations, systems, and control processes from the individual combatant and platform level to brigade levels. The OOS accurately and effectively represents specific combat, combat support, combat service support, and command, control, communications, computers, and intelligence activities. "OOS provides a complete simulation environment that supports the entire simulation life cycle from simulation and model development through scenario generation and execution to after-action analysis and review," said Tom Radgowski, program manager for OOS Architecture and Integration. "To meet diverse domain requirements, OOS is developed as a composable line of individual products. Users can combine different products within the OOS product line to meet their individual needs."

As an example of what composability provides to the user, Surdu asked Team OneSAF how the peer-to-peer architecture could be scaled to handle hundreds of thousands of simulated entries. "They told me that the network services layer was architected to allow the simulation to operate in its peer-to-peer mode or it could be run on a single multiprocessor server with shared memory," said OneSAF project manager, U.S. Army Lt. Col. John Surdu. "Due to their layered architecture of OneSAF, this differ-

ent mode of operation would be completely transparent to the rest of the simulation."

The OOS is designed for use across the three Army modeling and simulation domains: Advanced Concepts and Requirement; Training, Exercises and Military Operations; and Research, Development, and Acquisition. "The requirements for this simulation were that it meet the needs of sophisticated analysts who need high fidelity – as well as staff trainers – who need low

organization. The U.S. Army Program Executive Office for Simulation Training and Instrumentation awarded a series of task orders to hand pick a set of contractors that could best build the individual pieces of the OOS. They also enlisted on-site representation from the end-user community and reach-back access to a wider group of users for the development process.

"The OOS development effort is characterized by an unparalleled level of cooperation between the government team and the various contractor teams working on the program," said Radgowski.

Science Applications International Corporation (SAIC) served as the OOS Architecture and Integration Task Order lead, and established a comprehensive process set for software and system development. "Our processes are tailored from general SAIC processes that have been externally certified as Capability Maturity Model® Level 4," says Radgowski. "OOS processes are documented in a Web-based electronic process guide that is available to all OOS developers. Compliance to these processes is monitored by independent quality assurance audits and tracked by software development metrics. Peer reviews occurred at all phases of the development to ensure timely defect prevention and optimal product quality. Our metrics indicate that these reviews identify more than 90 percent of all defects."



**"Peer reviews occur at all phases of the development ... Our metrics indicate that these reviews identify more than 90 percent of all defects."**

fidelity and high entity count," said Surdu. "Team OneSAF has done an exceptional job of creating a scalable, flexible, extensible, composable architecture that is technically the best simulation architecture I have seen in several years of working under the hood in military simulations."

## Team OneSAF Structure

The Team OneSAF approach incorporates government managers, contractor development teams, and end users into a single

## An Integrated Environment

The core development team is collocated in a single facility and is supported by the OOS Integrated Development Environment (IDE). The IDE is a comprehensive Web-based management and development environment that enables an efficient and effective interchange of ideas and concerns, and facilitates the swift resolution of issues as they occur.

The IDE also provides support services for OOS participants (such as beta site testers) who participate in the program at geographically diverse locations. Access to the IDE is provided throughout the OOS Web site <[www.onesaf.org](http://www.onesaf.org)> providing

access to numerous tools that help manage action items, peer review artifacts, problem trouble reports, risks, and help desk requests. The BuildBoy application routinely builds and automatically regression tests new OOS software, and publishes build results and status on its Web site.

The Web site is configuration managed, which enables secure, distributed development. The IDE capabilities are a combination of commercial off-the-shelf (COTS) products, custom-developed products, and customized configurations of COTS products, and represent an open network architecture that is capable of scaling large numbers of development machines, rapidly introducing new resources and providing a stable, secure development environment.

### OOS Quality Build Methods

The OOS IDE provides automated tools to collect and report technical and management metrics that are reviewed on a regular basis using a formally defined Quantitative Process Management and Software Quality Management process to support the OOS' formal metrics plan. Bi-monthly meetings are held to analyze trends and identify areas where improvements can be made. This allows program management personnel to drill down and examine productivity or quality issues in detail, according to Radgowski.

The OOS is built using a spiral development methodology and extreme programming (XP), and is designed to be hardware-platform and operating-system independent. The developers build and integrate their software on Windows, Linux, and Solaris systems and formally test the results after every development spiral.

"The OOS requirement to integrate a significant portion of directed reuse components into the end product is enabled by the application of XP concepts," said Radgowski. For example, the OOS uses a succession of small, rapid, build cycles to integrate frequent releases, therefore avoiding the problems of a single integration. The process begins with overall four-block (A, B, C, D) planning: a development process where user feedback is incorporated into the final product and tested at sites across the country. Currently, Blocks A and B have been distributed to select organizations within the Army, Navy, and Marine Corps.

Block A was developed to be an initial implementation of the OOS architecture with the corresponding tools, components, and services to allow it to execute entire simulation life cycles. Block B contains a comprehensive set of current OOS components, including the system, unit, entity, and behavior composers; the command, control,



*Team OneSAF is characterized by unparalleled cooperation between government and contractors.*

communications, computers and intelligence adapter; the military scenario development environment; the 3D viewer; the after action control component; the environmental runtime databases; the data repositories; and the initial software application compositions for execution.

"Each of these four blocks is deployed to selected sites for evaluation and comment," said Radgowski. "We provide a user feedback tool so that beta site evaluators can provide comments back to the development team. Senior OOS staff individually evaluate each comment ... If they find a bug, or give us insight on how to make OOS better, we can react very quickly to their comments."

The process begins with overall block planning, which determines the goals for the respective block, and allocates the goals into eight-week builds for individual software development teams. Each team performs detailed planning for a given build four weeks prior to the beginning of the build. Each build contains requirements analysis, design, code and unit test, and software integration phases. Once a development team completes these phases, it formally hands its software to the Integration and Test team, which conducts an independent test of the code. If the code passes this test, it is nominated to the Test Working Group for designation as a user assessment baseline (UAB). If approved, the UAB is then made available for user evaluation and demonstration. This continuous integration process helps ensure that independently developed OOS components remain in sync.

"The use of radical programming has been of significant benefit. Every eight weeks, the program executed Integration and Test and a review of the current state of the software by engineers and, most importantly, the users," said Gregory Miller, senior engineer, Alion Science and Technology, support to TRADOC Project Office OneSAF.

Because of the combined programming methods, any upgrades or fixes are promptly made and the engineers get immediate feedback from users on how to make the

system better. "Team OneSAF is manned by people recognized as gurus within the modeling and simulation field. The composable architecture ... creates a unique solution and has made fans of skeptics," said Surdu.

### Cost

The OOS program was delivered extremely close to cost and time estimates. Since the software is designed to allow all users to interact on the same software, the government only has to maintain one system instead of several. Implementing standardization methods also saves time in training and sharing of information.

One impressive recommendation for the software is the expressed desire for other major programs such as the Marine Corps Combined Arms Staff Trainer, and the Army's Synthetic Theater of War and Program Executive Office for Simulation Training and Instrumentation Common Gunnery Architecture programs to use the OOS software in their development efforts. As well, the Army's Future Combat Systems program has designated OOS as a training common component.

The Army's investment is already paying off. "On quarterly earned value reports, it is amazing for programs to be within 1 percent cost and schedule variance," said Surdu. "Typically the OOS team is within 0.5 percent — and the program has *never* been restructured. I have a strong technical background, but the engineers working on OOS amaze me daily with the strong technical decisions they make. The contractors working on Team OneSAF take the long-term view to make sound technical decisions that are right for the customer."♦

### Project Point of Contact

**Tom Radgowski**  
**SAIC**  
**12901 Science DR**  
**Orlando, FL 32826-3014**  
**Phone: (321) 235-7739**  
**E-mail: [tradgowski@ideorlando.org](mailto:tradgowski@ideorlando.org)**

## Patriot Excalibur Software Enables Full-Scale Deployment of Battle-Ready Units

Chelene Fortier-Lozancich  
CROSSTALK

*Whether during wartime deployment or peacetime record keeping of troops, being prepared for any eventuality is of utmost importance. Software that enables this war fighter preparation must be a simple tool set that encompasses all needs, a one-stop-shopping experience that can handle scheduling troop activity, provide easy access to needed data, and be a usable product that does what is needed.*

Preparation for a wartime deployment often begins during peacetime. It can mean the difference between an organized, battle-ready unit and a unit that is still trying to schedule troops, order supplies, and prepare for battle. During a major deployment such as Operation Iraqi Freedom, currently under way, a disciplined tool set for scheduling aircraft, people, training, and flying must be in place. This tool set must also, in addition to providing all necessary information, be a *one-stop-shopping* experience that can handle day-to-day tasks, easily provide access to data, and combine these and other combat-capable assets together to provide the U.S. Air Force the ability to meet any threat.

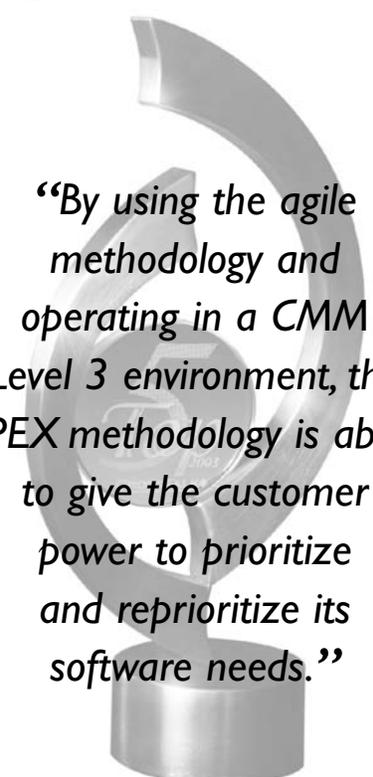
All this does not matter if the product sits on the shelf.

Patriot Excalibur (PEX) is software that is built from the bottom up with the warfighter in mind, according to PEX group lead Linda Crabtree. "The software is developed *for* the user – that was a core requirement. A product that does everything, but not in the way the user wants it done, sits on the shelf. Our product does not sit on the shelf."

The need was simple: Create a squadron-level automated environment where data entered in one application is usable throughout others, during both peacetime and wartime. The software must also keep the user in the loop, provide single- and multi-crew scheduling, enable single input of data, be interoperable with external systems, and be cost efficient. "We provide a squadron-level, PC-based tool set that helps aircrews conduct effective, timely operational tasks in an integrated, standardized system of products that connects the functional areas across the squadron," says Crabtree.

PEX's major design goals were to create a product based on commercial Microsoft standards, be configurable by individual units, be interoperable with

external systems and databases, and derive its program requirements by keeping the user in the development loop. "This is the first software [squadron automation package] I have seen that was tailored to meet all facets of a flying squadron's operations," says Master Sgt. Jeffrey Crawford, operations superintendent.



**"By using the agile methodology and operating in a CMM Level 3 environment, the PEX methodology is able to give the customer power to prioritize and reprioritize its software needs."**

### An Organized Squadron

Squadron-level aircrew and aircraft scheduling during wartime deployment is usually accomplished with a different set of software tools than those used by the same unit at home during peacetime, according to Crabtree. "Learning a different set of tools during wartime deployment compounds the difficulty of an already stressful situation. It also adds to the *spin up* time needed to get ready, as a lot of raw data such as aircrew names, qualifications, aircraft data, and scheduling

parameters need to be input into the different scheduling systems."

Previous squadron automation software was adequate for basic aircrew scheduling functions, but was limited and did not offer the broad range of functions that defines PEX. "Previously, our aircrew entered all of their meetings or appointments into a Microsoft Excel spreadsheet or placed them on a dry-erase board," says Crawford. "Flight commanders had to print out this spreadsheet or consult the board in order to coordinate the weekly flying schedule."

PEX has the ability to interface with the Aviation Resource Management System (ARMS), providing several benefits, according to Crawford. PEX provides squadron decision makers with real-time aircrew training currency information; flight managers can pull currency information from the ARMS into PEX on a daily basis; the aircrew can view their training status from their desktop, and no longer have to request flight management personnel to run an ARMS report that shows what events are coming due.

PEX can also interface with the Core Automated Maintenance System, giving a squadron the ability to have real-time aircraft status on a desktop, something that was lacking from previous squadron packages. "From an aircrew scheduler's point of view," says Crawford, "PEX makes the job much easier. It provides one source for all of the information required to successfully run a squadron's flying operation."

### Agile + CMM

Perhaps one of the most important aspects of PEX is the way it was developed – using not one development approach, but a combination of extreme programming (XP) or agile methodology, and the Capability Maturity Model® (CMM®). When PEX team first started looking into using XP, they agreed it was a great concept but given their current development environment did not think it

would be possible to incorporate. They were also concerned about maintaining their CMM Level 3 status.

A year later, PEX development team found themselves facing several obstacles: an inaccuracy in estimating development time, using antiquated techniques that did not fit their object-oriented development, and rapidly changing requirements. The development team was facing late delivery, overtime, and team turmoil.

After revisiting their methodology, a team member suggested looking at agile development, which is a collection of values, principles, and practices for software that can be applied to a development project. The team mutually agreed to revisit this development method, and from the beginning, agile/XP became the preferred method. After several meetings, the PEX development team unanimously agreed to make a change, but wanted to keep their CMM Level 3 processes. They decided that agile development was the method that would best accommodate the team's needs.

"The actual results of the change have exceeded my expectations," says Crabtree. "The stability of the product has increased dramatically, the amount of work accomplished has increased, the team is genuinely enthusiastic about coming to work, our estimation ability has improved and continues to improve, and our user numbers are increasing at an astounding rate."

By using agile methodology and operating in a CMM Level 3 environment, PEX methodology is able to give the customer the power to prioritize and reprioritize its software needs. "PEX team receives its requirements directly from its users," said Kelly Goshorn, PEX program manager. "This is a tremendous timesaver for the development team as far as understanding the requirements. PEX (team) also uses XP with two-week iterations and a six- to eight-week update, allowing functionality to continually be released to the field."

In the three months following the release of PEX v.3.2 in September 2003, the PEX office received three problem reports and 78 enhancement requests. PEX provides monthly metrics to the customer, and the software management tool VersionOne provides insight into individual member load, team load, and number of stories in each iteration and release.

PEX is a three-tier object-based client/server application that is written using Microsoft Visual Studio. The interfaces to the database use Active Data Object and Object-Linking and



*The Patriot Excalibur structure interfaces easily with other service systems.*

Embedding Database. User input is entered via the graphical user interface. With the release of v.3.2, PEX began providing Web-enabled modules. Several of these modules are both desktop and Web-enabled. PEX can also provide data to other applications from the database via Web services using Extensible Markup Language. PEX runs in a Windows NT Version 4, Windows 2000/XP environment. Due to the volatility of the requirements and the need for better estimation, PEX development has benefited tremendously from agile/XP development.

"116ACW was able to reorganize efficiently based on PEX," said Maj. Thomas J. McNeill, Wing PEX administrator and senior director. "It is impossible to determine the number of saved missions and hours of coordination and senior-level questions attributed specifically to the wing-wide implementation of PEX."

### Cost Effective for the User

PEX is cost-effective for the user. Since the product is government-owned software developed by a contractor under a time-and-materials contract, it means Department of Defense personnel who have access to a <.mil> account can use PEX off the shelf at no cost. "Our customers have access to our team around the clock via our Web site, help-desk hotline, pexhelp mailbox, and feedback forms," said Crabtree. "We provide in-house training classes at no charge and send our operation specialists out to their site for the cost of the temporary duty only. We provide user support in a timely fashion at no charge."

"PEX was a significant increase in Nellis's [Air Force Base] scheduling capability with significant reduction in man-

hours," says 57WG Chief of Scheduling Maj. Charles Blank. "The entire team of PEX has been infinitely helpful. They have responded to every input Nellis users asked for and continue to provide outstanding support through the help desk."

Since the release of PEX v.3.2, its growth rate has steadily increased. Its user base has grown from 12 units two years ago to more than 100 today. One unit was commended for *looking to the future* for implementing PEX throughout the squadron, while another unit using PEX got an outstanding on their operational readiness inspection, with major credit going to PEX's Web usefulness. The PEX program is being embraced by squadrons all over the country, according to Goshorn. "There are approximately 100 units using PEX currently. The continuity gained for the units by using the same software is astronomical. A user can communicate from one squadron to another and be familiar with the product when arriving on station."

With all this functionality, PEX is a valuable asset to the Air Force, providing the warfighter with needed tools to meet any threat. "The only limiting factor of PEX," says McNeill "is the fact that not everyone is using it."◆

### Project Points of Contact

**Kelly Goshorn**  
46 TW/XPI  
Eglin AFB, FL 32542  
Phone: (850) 882-2358  
kelly.goshorn@eglin.af.mil

**Linda Crabtree**  
Phone: (850) 882-2348  
linda.crabtree@eglin.af.mil



*(Photo above) Representatives from the Top 5 projects accepted the awards at the 2004 Systems and Software Technology Conference in Salt Lake City. They were, back left: Kelly Goshorn, Lt. Col. James Chapman, Col. Ralph Sees, Thomas Radgowski, Marty Montgomery, Beverly Kitaoka. Front row from left: Stephen Lutz, Harla Cynthia Inteso, Peter Nolte, Linda Crabtree, Lt. Col. John Surdu. (Photos below) Col. Ralph Sees and Cynthia Inteso speak as part of the Wednesday plenary session.*

## CROSSTALK Presents Top 5 Awards at the Systems and Software Technology Conference

The winners of CROSSTALK's 2003 U.S. Government's Top 5 Quality Software Projects were presented with their awards at the 2004 Systems and Software Technology Conference (SSTC) held recently in Salt Lake City. Peter Nolte of the Office of the Under Secretary of Defense, Acquisition, Technology, and Logistics, the department sponsoring the contest, presented individual awards.

Top 5 judge Dr. David A. Cook, senior research scientist, Aegis Technologies Group, Inc., introduced the representatives from each winning project to a plenary session that pulled its

audience from more than 2,200 SSTC attendees. Following the presentations, a member of each winning project briefly presented at the plenary session and answered attendees' questions after-

wards.

Throughout the U.S. government, many organizations are using processes and practices that result in the successful delivery of projects with significant software content. This includes using well-defined and proven processes and practices to develop, manage, and integrate software. The intent of the U.S. Government's Top 5 Quality Software Projects' search was to recognize the outstanding performance of these software teams and to promote their efforts at best practices.

This was the third year for the competition. ♦





... row from  
... Kooima,  
Receiving the award for One SAF Testbed Baseline (above) were Beverly Kitaoka (left) and Thomas Radgowski (right), presented by Peter Nolte (center). (Photo below) Harlan Kooima (left) responded to a question asked by an audience member during the question and answer portion of the presentation.



## Top 5 Software Projects Scoring Criteria

Reviewers from the Software Technology Support Center (STSC), Hill Air Force Base, Utah, used the following criteria and point system to score all nominations as part of the process to select the 2003 U.S. Government's Top 5 Quality Software Projects finalists. Each nomination was awarded points (up to a maximum value) based on how well the project performed within each category: customer value, performance, technical value, and reviewer's discretion. At least three STSC consultants/engineers scored each nomination with the top one-third of the nominations closely scrutinized by the internal board to select the finalists.

### Customer Value – Maximum 40 Points

#### Problem Reports

- Were responses to the problem reports and questions timely?

#### Value

- What was the measured value to the customer's mission (return on investment)?

#### Benefits and Satisfaction

- Is the end product useable?
- Is the customer satisfied with the end result?
- What other benefits were provided to the customer?
- Was the developer collaborative?
- Did the developer listen to the customer?
- Was the developer knowledgeable? Informative? Helpful?
- Was the developer professional in letting the customer know requirements trade-offs?

### Performance – Maximum 25 Points

- Did the developer meet the contracted schedule?
- Did the developer meet the contracted budget?
- How many problem reports have been written against the product since system test?
- Is the customer satisfied with the performance?

### Technical Value – Maximum 20 Points

- Was the problem challenging? How hard was this project to implement?
- Was the solution innovative? What approach was used to solve the problem? What technical value did they provide to the world?
- Is the project reusable? Can someone else use the end product, portions of the end product, code, process, or the product's technology to solve a future government problem?
- Is the project repeatable? Given a similar problem, could the group repeat this success or were they just lucky this time? (Did they use defined processes, trained people, etc.?)

### Reviewer's Discretion – Maximum 15 Points

Use or don't use these points as discretion dictates. Suggested considerations include the following:

- Previous awards. (CMM, ISO 9000, Malcolm Baldrige, etc.)
- Customers. (Will one small organization use this or will it be dispersed worldwide?)
- Do they have measures that can be used for oversight and additional improvements?
- What is the atmosphere/morale of the developing organization?

## CROSSTALK Honors the 2003 Top 5 Quality Software Projects Finalists

Pamela Palmer  
CROSSTALK

*It was difficult to narrow the field from the many successful government projects entered in the third annual U.S. Government's Top 5 Quality Software Projects contest. As a result, the following 10 projects are being honored as 2003 Top 5 Finalists.*

### Center Ops OnLine

**Customer:** Air Force Materiel Command

The Center Ops OnLine (COOL) v3.0 is an enterprise ops desk automation application <<https://cool.edwards.af.mil>> used at seven Air Force bases across the country. The COOL is Web-based and allows authorized users access via the Internet. It allows users to efficiently manage aircrew testing, flight crew information files, aircrew currency, flight authorization, and aircrew training. The COOL application makes aircrew readiness information available in one place, incorporates the Air Force Materiel Command (AFMC) Operations Group regulatory requirements, works with several other Air Force systems, minimizes data entry, and maximizes data currency. The COOL v3.0 supports all AFMC bases and could expand to cover every Air Force base in the continental United States or overseas. Please e-mail questions to <[COOL.info@edwards.af.mil](mailto:COOL.info@edwards.af.mil)>.

### Deliberate and Crisis Action Planning and Execution Segments

**Customer:** United States Air Force

The design, development, fielding, and operational use of the Deliberate and Crisis Action Planning and Execution Segments (DCAPES) to execute Operation Enduring Freedom and Operation Iraqi Freedom, culminates the most sweeping changes for how the Air Force projects air power in over 20 years. The DCAPES provides near real-time integrated command and control, planning, and execution monitoring information to Air Force functional users in operations, logistics, manpower, and personnel, providing a single integrated planning environment. With DCAPES, Air Force planners can rapidly and accurately identify and source personnel, equipment, and sustainment capabilities to meet the combatant commander's operations plan requirements. Addition-

ally, the DCAPES enables senior Air Force decision-makers to rapidly adjust operations plans to accommodate ever-changing scenarios. The evolution of DCAPES is swiftly replacing old stove-piped, domain-centric systems by producing a single, fully integrated, replicated database. The DCAPES has been assessed at a Capability Maturity Mode® (CMM®) for Software Level 2 and the program is pursuing a CMM Integration<sup>SM</sup> Level 3 rating.

### F-15 Bench Top Reconfigurable Automatic Tester - Test Program Sets Rehost

**Customer:** Ogden ALC Electronics Division

The test program sets (TPSs) delivered are used to test and repair the shop replaceable units (SRUs) in the F-15 generator control unit on the Benchtop Reconfigurable Automatic Tester. These TPSs will functionally test the SRU and specify the faulty components; the user then replaces those components and retests the SRU. After it proves to be a good SRU, it is returned to supply. The TPSs consist of test program software; interface test adapter, including associated hardware; engineering drawings for the developed hardware; technical manuals, which include test procedures manual (TPM); operation and maintenance (O&M) manual; and software design documentation. The TPM contains instructions for performing functional and diagnostic testing of the SRU. The O&M manual contains operation and maintenance instructions with an Illustrated Parts Breakdown for the interface test adapter and associated hardware. Software design documents contain additional detailed engineering test information helpful in operating the TPSs. The success of this development is based upon technical expertise being systematically applied to solve technical, logistical, and managerial problems according to a well-defined TPS development process.

### FireFinder Q37

**Customer:** U.S. Army, Project Manager FireFinder

The AN/TPQ-37, or FireFinder Q37 (FF Q37), is a phased array, pulsed Doppler, S-Band radar developed for counter-battery artillery detection and location. The system identifies the exact location of enemy units that fire upon friendly forces, before the bullets or rockets ever land, and provides for the coordination of returned fire in a matter of seconds. There are approximately 200 FF Q37 systems fielded to the U.S. Army and Marine Corps worldwide. A *digitalization* software upgrade lets a modern Intel-based workstation be integrated into the FF Q37. This allows modern digital communications, standard National Imagery and Mapping Agency products for height correction, and the incorporation of a graphical user interface. The most recent version of FF Q37 has been deployed during 2003 to support Operation Iraqi Freedom. The FF Q37 was one of four representative systems appraised that led to the Communications Electronics Command, Software Engineering Center, Fire Support Software Engineering at Ft. Sill being one of the first Department of Defense affiliates to achieve Capability Maturity Model® Integration Level 5.

### Forward Observer System

**Customer:** U.S. Army, Project Manager Intel Effects

The Forward Observer Systems (FOS) is fielded to provide field artillery forward observers with the capability to direct and coordinate field artillery, mortar, close air support, and helicopter munitions onto targets; to provide commanders, fire support officers, fire support teams, and forward observer and survey teams with the capability to plan collective actions through maneuver and artillery graphic map displays; to provide for storage of survey calculations and control points by field artillery commanders; and to provide a message set for use by survey and fire support teams. The FOS is a combat-critical

system for the U.S. Army and the Department of Defense (DoD) that provides an interface for first contact with the enemy. Before the enemy knows where U.S. troops are, the FOS is used to coordinate enemy location with U.S. firing units. The FOS has been upgraded and deployed to support Operation Iraqi Freedom during 2003. The FOS was one of four representative systems appraised that led to the Communications Electronics Command, Software Engineering Center, Fire Support Software Engineering at Ft. Sill being one of the first DoD affiliates to achieve Capability Maturity Model Integration Level 5.

### **Global Combat Support System** **Customer: Special Program Office**

The Global Combat Support System – Air Force (GCSS-AF) delivers network-centric enterprise services through a suite of commercial off-the-shelf (COTS) products integrated under a common COTS security layer. These core enterprise services provide a common software and hardware infrastructure for the Air Force to integrate, in some cases eliminate, and then operate its more than 640 combat support systems. During 2002-2003, there were more than 90,000 system users or 350,000 Web pages served per day. Availability has been approximately 99.5 percent, or about an hour of both scheduled and unscheduled downtime every 10 days. The GCSS-AF supports the warfighter at 30 locations in southwest Asia. On day one of Operation Iraqi Freedom, 14 key supply chain capabilities were secured and put back into production within six hours and made available to users worldwide. Those capabilities are now the second most heavily used applications on the Air Force Portal. Work on the GCSS-AF was done at Warner Robins Air Logistics Center's Software Engineering Division, Section E, Avionics Test Program Branch, Maintenance Directorate.

### **Marine Corps Total Force System** **Customer: U.S. Marine Corps and the Defense Finance and Accounting Service**

The Marine Corps Total Force System (MCTFS) is an integrated pay and personnel system for all active, reserve, and retired Marines. The U.S. Marine Corps and the Defense Finance and Accounting Service jointly own the system. The MCTFS is one of the largest automated information systems and the only integrated pay and personnel system that is fully operational capable within the Department of Defense. System revisions originate from congressional legislation, new policy, cost savings initiatives, and modifications to existing

functionality, and are formally prioritized by a joint Configuration Control Board. These system changes are then bundled into semi-annual software releases and scheduled into overlapping 10-month development cycles. The system is used to manage more than 498,000 Marine records for active, reserve, and retired members. The MCTFS processes in excess of 17 million transactions yearly, and computes an average gross payroll of \$238 million per semimonthly pay period totaling \$5.712 billion in payments annually. Transactions processed by the MCTFS can be generated in stand-alone, client/server, and Web-based environments, including users not connected to a network in a remotely deployed location. The MCTFS was formally assessed at Capability Maturity Model® for Software Level 3 in 2000, and currently employs processes consistent with Level 4.

### **Navy Standard Integrated Personnel System**

#### **Customer: Program Executive Office for Information Technology**

The Navy Standard Integrated Personnel System (NSIPS) is an automated information system that delivers field-level pay and personnel data to update corporate databases in peacetime as well as during recalls, and during both a partial and full mobilization. Most importantly, the NSIPS collects, passes, and reports timely, accurate data on active and reserve members in the continental United States, overseas, and aboard ships. The NSIPS also provides the ability to send and receive work items, updates, and records to and from ships that do not have the ability to maintain direct connection to the main server via a secure Internet connection at all times. The Web-enabled NSIPS is a centrally hosted implementation with primary access from Web-browser client terminals. It eliminates four legacy field input systems. The system supports approximately 500,000 records serviced by 8,000 Navy personnel and pay specialists at nearly 572 Personnel Service Activities, Personnel Service Detachments, and Navy Reserve Activity sites. The NSIPS is now deployed to 80,583 reservists and 400,956 active duty personnel.

### **Tactical Tomahawk Weapons Control System**

#### **Customer: PMA 282 (U.S. Navy)**

The Tactical Tomahawk Weapons Control System (TTWCS) provides surface ship on-board software and attendant hardware and submarine on-board software to plan and control the launch of Tomahawk cruise missiles. The TTWCS development is part of the Tactical Tomahawk Weapons System

Upgrade to improve the flexibility and responsiveness of Tomahawk cruise missiles, to add new capabilities, and to provide an upgrade for existing fleet systems. The TTWCS includes the capability to receive electronic tasking via legacy communications interfaces, to reduce engagement planning time due to increased automation, and to perform launch platform mission planning, which allows surface ships and submarines to plan global-positioning-system-only missions onboard, thereby improving tactical responsiveness. The software is executed by operators at four tactical display consoles on surface ships and from one to four consoles on submarines remotely. The TTWCS interfaces with several shipboard systems, including the ship's navigation system, weapon vertical launch system, Global Command and Control System–Maritime, and communications networks. The TTWCS employs processes for Capability Maturity Model® Integration Level 5 for Systems Engineering/Software Engineering.

### **Wide Area Augmentation System**

#### **Customer: Department of Transportation – Federal Aviation Administration**

The space-based Wide Area Augmentation System (WAAS) provides an augmentation of the global positioning system (GPS) to supply the accuracy and integrity for the civil signal required to support safety-of-flight applications. The WAAS provides pilots with the data to navigate without additional navigation aids, for both en-route and Lateral Precision with Vertical Guidance. The WAAS is also used by land- and sea-based enterprises needing accurate positional information such as surveyors, farmers, and maritime users. The WAAS software consists of four computer software configuration items (CSCIs). The Data Collection Processing (DCP) CSCI operates at 25 U.S. sites. The DCP receives data from the GPS and geostationary earth-orbiting satellites, selects data of interest, and forwards that data to the Correction and Verification CSCI, which is the algorithmic center of WAAS. It computes correction and integrity data to be sent to the GEO Uplink Subsystem Processing CSCI for forwarding on to the WAAS receivers in aircraft or on the ground. The WAAS is the first system of its kind certified for use by the Federal Aviation Administration. The WAAS program was a major component in the Capability Maturity Model® Integration Level 5 rating received by Raytheon in Fullerton, Calif. ♦



# Using Software Metrics and Program Slicing for Refactoring

Dr. Ricky E. Sward  
U.S. Air Force Academy

Dr. A.T. Chamillard  
University of Colorado at Colorado Springs

Dr. David A. Cook  
AEGIS Technologies Group, Inc.

*Refactoring can improve the quality of a software system as measured by coupling, cohesion, and cyclomatic complexity, but knowing which refactoring choices should be implemented is key. This article presents an approach that guides the refactoring of software systems by combining the use of software metrics and a technique called program slicing. Program slices produced from a single software module are sorted by the respective values of the metrics; a design that provides the most beneficial metric values can be selected from these. This approach can produce a software system with higher quality and maintainability as measured by the metrics.*

Many systems designed today have very long life cycles, especially in the military. Often, a software program is expected to perform for many years, and undergo frequent updates and requirements changes. Large-scale software systems are prone to quality problems [1] during development. Constant changes to existing systems only lead to additional quality problems.

One way to help control defects and reduce high maintenance costs is to use refactoring. Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves its internal structure [2]. Refactoring is an option during both the development and maintenance phases. Unfortunately, refactoring the design can be very resource intensive, and automated tool support is considered crucial [3].

This article presents an approach for automating a large part of the evaluation and refactoring process. By combining the use of software metrics and a technique called program slicing, the refactoring process is guided toward a design with higher quality and more maintainability. First, we discuss how several different software metrics can be used to evaluate software quality and the effects those metrics have on defects, testing effort, and maintenance cost. We then discuss how program slicing can use those metrics to guide design-refactoring decisions. The final section presents our conclusions.

## Software Metrics

We use software metrics to try to quantify particular characteristics of software systems, such as quality, maintainability, or reliability. In general, however, these characteristics cannot be measured directly. Instead, we directly measure particular attributes of software by using

software metrics and then infer information about quality from those direct measurements [4].

Three commonly used software metrics are coupling, cohesion, and McCabe's Cyclomatic Complexity [5]; all three have been extended from their original definitions for use with object-

---

**“One way to help control defects and reduce high maintenance costs is to use refactoring.”**

---

oriented systems (OOS). In this article, we discuss our ideas in the context of a system that was implemented using structured design techniques, though the process could also be extended for use with OOS.

The first metric to consider is coupling, which measures the strength of the connections between the software modules that comprise a particular system to quantify the dependencies between the modules. The key idea is that the more interdependent the modules in the system are, the more difficult the system is to understand and the more likely it is that changes to one module will affect other modules in the system.

Yourdon [6] originally described several different kinds of coupling, including data coupling, control coupling, hybrid coupling, and so on. McConnell [7] has updated coupling to include classes of coupling. In the technique described in the following section, we only consider data – or normal – coupling. In other words, the main focus in terms of coupling is on the information

that flows between the modules in the system. We measure this coupling by counting the number of parameters (i.e., pieces of information) passed into and out of each module.

As Yourdon points out, “The coupling between modules in tentative structural designs can be evaluated to guide the designer toward less expensive structures.” Our idea is to provide precisely this kind of guidance, but to do so with extensive automated tool support. This guidance would be useful in both the design and the maintenance phases, though we believe most refactoring occurs in the maintenance phase.

The second metric to consider is cohesion, which measures how strongly the elements of each module are related to each other. Cohesion was originally defined in an article by Stevens [8], and the concept has been updated as programming languages and their capabilities have evolved. McConnell [7] contains a working definition of the current classes of cohesion. At a high level, a module with high cohesion accomplishes a single function using only the data required to accomplish that function. As with coupling, Yourdon defined multiple levels of cohesion, though we limit our interest to functional cohesion in this article.

Coupling and cohesion are related, though not perfectly correlated. As we increase the cohesion of the modules in the system, we tend to reduce the coupling between those modules. This is an important consideration, because although researchers have proposed various ways to measure cohesion [9], it is much more difficult to measure than coupling. In this approach we do not measure cohesion directly, but instead rely on the relationship between coupling and cohesion to infer information about the quality of a module. If we find

that we also need to directly measure cohesion to make our approach more effective in practice, we can extend the metrics calculated by the tool to include cohesion as well.

The final metric, which is probably the most commonly used metric of those discussed here, is McCabe's Cyclomatic Complexity. At the module level, this metric is the number of linearly independent paths through the module. Modules that contain many possible paths are more complex than those with fewer paths, so as the cyclomatic complexity of a module increases, so does its complexity. We note that this metric is equal to one more than the number of decisions contained in the module.

Based on the discussion above, our approach uses coupling and cyclomatic complexity metrics as described in the following section. The coupling metric provides insight into the interaction between the modules in the system, while the cyclomatic complexity metric gives insight into the complexity of each individual module. Remember that we do not directly include cohesion, though we could extend our approach to do so if it proves to be helpful in practice.

As stated in the introduction, our goal is to use software metrics to provide guidance to those undertaking refactoring efforts. It is important to note, of course, that we do not refactor code simply for the sake of better code; rather, we expect some return on the investment expended on any refactoring efforts. We must therefore consider some of the important relationships between our software metrics and software quality, testing costs, and maintenance costs.

Intuitively, we expect software with high coupling and low cohesion to be of lower quality than software with low coupling and high cohesion. The additional programmer effort required for understanding highly interrelated modules and their effects on each other leads to a higher potential for mistakes. Similarly, a programmer working on a module with low cohesion needs to keep track of multiple functions being performed by the module rather than on a single function, which also increases the potential for programmer error. Our intuition turns out to be true in practice. Research on operational systems has shown that modules with high coupling/low cohesion contained seven times as many errors as modules with low coupling/high cohesion [10]. In addition, programmers spent almost 22 times as many hours correcting the

errors in those modules with high coupling/low cohesion. Clearly, coupling and cohesion have a significant impact on both the quality of the software and the effort required to fix errors in the software.

We also expect all three metrics to have an impact on the amount of effort associated with software testing. Because coupling measures the dependencies between modules, higher coupling implies the need to expend more effort accomplishing integration testing of the modules. Modules with low cohesion implement more than one function; testing the functionality of that module (typically during unit testing) requires more test cases to cover all of that module's functionality. Cyclomatic complexity essentially measures the number of paths through a module, so modules with higher cyclomatic complexity will require more test cases to cover all the paths.

---

***“The coupling metric provides insight into the interaction between the modules in the system, while the cyclomatic complexity metric gives insight into the complexity of each individual module.”***

---

Discussions of software quality and testing effort apply both to the original development of a system and maintenance of that system. We are also concerned, of course, with the cost of maintaining systems. Research shows that we can account for more than half of the variability of maintenance productivity by taking cyclomatic complexity into account [11]; in other words, we can significantly improve our estimates of maintenance costs through consideration of the cyclomatic complexity (and lines of code) for the system to be maintained. Perhaps even more importantly, we know that modules with higher cyclomatic complexity are more difficult to maintain, so if we can reduce the complexity of the modules we can reasonably expect a corresponding reduction in

maintenance costs.

It is clear that refactoring software to improve coupling, cohesion, and cyclomatic complexity of the software yields improvements in overall software quality and reductions in testing and maintenance costs. Despite the clear benefits associated with refactoring, the amount of effort required to refactor large systems without tool support is generally prohibitive [3]. Metrics and other methods have been proposed to help guide program refactoring [12, 13, 14, 15]. One problem with traditional metrics is that they are often not useful for making fine distinctions between routines and modules [7, 16]. Refactoring does not have this limitation. The following section describes our approach for guiding the refactoring process through the use of program slicing and software metrics.

## **Program Slicing, Metrics, and Refactoring**

When considering options for refactoring, a technique known as program slicing can be used to isolate portions of a software system. A program slice is a projection of the behavior from a software module that is needed to produce a particular value in the module [17]. By slicing a particular variable or parameter in a software module, only the lines of code required to produce that variable or parameter are extracted from the module. The resulting lines of code can be built into a separate module with variables and parameters of their own.

For example, consider the Ada procedure shown on the left side in Figure 1 (see page 22). This procedure produces both the `Highest_Max` parameter and the `Lowest_Min` parameter. The Ada procedure shown in the upper right side of Figure 1 shows the program slice built by slicing on the `Highest_Max` parameter. The Ada procedure shown in the lower right side of Figure 1 shows the program slice built by slicing on the `Lowest_Min` parameter. Note that only those parameters needed to produce either `Highest_Max` or `Lowest_Min` are included in their respective program slices.

Program slicing is useful for refactoring software systems [18] because it isolates portions of the software. For example as shown in Figure 1, instead of a single procedure that produces both the `Highest_Max` value and the `Lowest_Min` value, we now have two procedures that each produce a single value. The challenge with using program slicing for

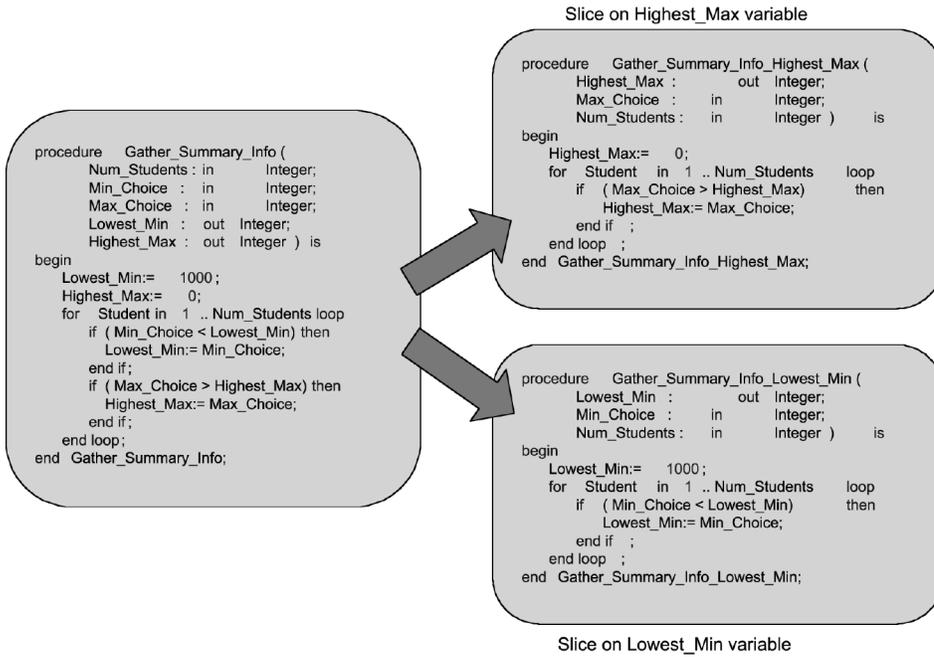


Figure 1: A Predictable Substation Assembly

refactoring is determining how to slice the software system properly in order to maximize maintainability and quality.

Since we want to refactor our software to improve coupling, cohesion, and cyclomatic complexity, software metrics can guide our choices when we use program slicing for refactoring. Each time we produce program slices, we can compare the values of the metrics from the original procedure to the resulting program slices. Clearly, program slices that produce better coupling, cohesion, and cyclomatic complexity are better than the original procedure and should be included in the refactored software system.

During refactoring, program slicing

may reduce the coupling between modules in the software system. For example, in Figure 1 the original procedure includes five parameters, but each program slice includes only three parameters. As we discussed previously, the number of parameters in a software module can measure coupling, so in this case, program slicing has reduced the coupling between modules.

Program slicing may also improve the cyclomatic complexity of a software module during refactoring. For the procedure shown on the left side of Figure 1, the value of the cyclomatic complexity metric is three. For each of the program slices, the value of that metric is two. In this case, refactoring using pro-

gram slices has resulted in software modules that have a lower value for the cyclomatic complexity metric.

Using program slicing for refactoring can therefore improve the quality and maintainability of software modules as measured by coupling, cohesion, and cyclomatic complexity. Admittedly, the example shown in Figure 1 is simplistic, but it demonstrates how using program slicing and metrics can guide the refactoring process.

### Slicing on Combinations of Variables

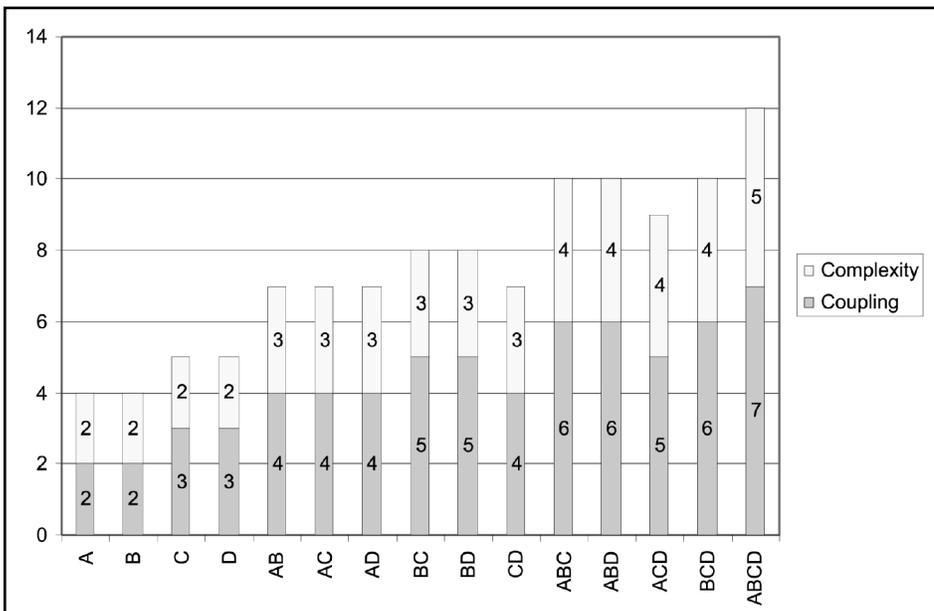
As is often the case, software systems contain modules that are much more complicated than the one shown in Figure 1. Software modules often have many different parameters (high coupling) and contain high levels of cyclomatic complexity. These modules present an opportunity to improve overall system quality by refactoring using the program slicing technique on different combinations of the parameters.

For example, consider a module that produces four values. For illustration, we will call the module Produces\_Four and call the four parameters A, B, C, and D. By using program slicing, we can build 15 different software modules, including the original module, from the possible combinations of these parameters. We can then calculate the coupling and cyclomatic complexity metric for each of the 15 modules individually.

Figure 2 shows the values of the coupling and cyclomatic complexity metrics for the 15 modules. The reader should realize that these are the values for program slices that were built from the Produces\_Four module that we used in our example. These values would differ for other program slices built from other modules depending on the code contained in those modules. Note that in Figure 2, the ABCD column represents the Produces\_Four module.

The following discussion shows how the information in Figure 2 helps to guide refactoring decisions. As we can see from the figure, slicing the original module into four separate modules for A, B, C, and D results in the lowest average coupling and cyclomatic complexity for the overall system. The average coupling and complexity of these four separate modules is lower than that of the original module, so breaking this module into four separate modules would be the best refactoring choice. The point is to lower the overall complexity of the sys-

Figure 2: Metrics for Program Slices



tem. By replacing the Produces\_Four module, which has high complexity and high coupling, with four new modules that have lower complexity and lower coupling, we can lower the average complexity and coupling for the system. We focus on the average of the metrics because we want to show that it will be easier to maintain the four new modules instead of one legacy module.

It could be the case, however, that organizational or management policies prevent you from selecting this option. For example, limits on the total number of modules in the system – or lower limits on the size of those modules – could preclude breaking the original module into four separate modules in our example. Effort should be expended to change such policies, especially when compliance will result in systems that are less maintainable than they could be. We also recognize, however, that some organizations will impose those policies regardless of the resulting impacts.

In this scenario, which refactoring option should you choose? In the following discussion, we assume that our policies constrain us to select exactly two modules in refactoring the original module.

As shown in Figure 2, the module for values CD has the same coupling and complexity as the modules for values AB, AC, and AD. All of these modules have lower coupling and complexity than the modules for BC and for BD. Selecting the module for AB along with the module for CD is a reasonable choice to replace the original module. The average coupling for this choice is 4 and the average complexity for this choice is 3. This option results in a lower average coupling and complexity for the overall system. It is also a better choice than selecting the module for AD along with the module for BC because the module for BC drives up the average coupling to 4.5. The values of the metrics for these program slices can help the software engineer select the best possible refactoring option that fits within the constraints placed on a software system.

Further analysis shows that the optimal choice in this situation is to choose the module for B along with the module for ACD. The average complexity for this choice remains at 3, but the average coupling is reduced to 3.5. This is a better choice than selecting the modules for AB and CD, since the average coupling and cohesion are less. Clearly, this is the best choice if the developer is constrained to selecting exactly two modules

in the refactoring process.

This illustrates how refactoring decisions can be guided by using program slicing and the values of metrics of the resulting program slices.

## Conclusion

Coupling, cohesion, and cyclomatic complexity have become accepted metrics for measuring the maintainability and quality of software systems. Refactoring can improve the quality of a system as measured by these metrics, but which refactoring choices should be implemented? We suggest using program slicing in conjunction with software metrics to guide the refactoring process. By slicing the software system

**“The return on investment in this refactoring process can be measured in lower error rates, fewer test cases per module, and increased overall understandability and maintainability.”**

on one or more variables, different refactoring options can be examined and evaluated using these metrics. The choices that program slicing provides can be sorted by the respective values of the metrics, and a design that provides the most beneficial metric values can be selected. It is the combination of program slicing and software metrics that guides the refactoring process.

A software system that has gone through this refactoring process has higher quality and is more maintainable. The return on investment in this refactoring process can be measured in lower error rates, fewer test cases per module, and increased overall understandability and maintainability. In both the design and maintenance phase, these advantages can be realized almost immediately.◆

## References

1. Jones, Capers. Assessment and Control of Software Risks. Prentice Hall, 1994.

2. Fowler, Martin. Refactoring: Improving the Design of Existing Code. Addison-Wesley, 1999.
3. Tahvildari, L., and K. Kontogiannis. “First International Workshop on Refactoring: Achievements, Challenges, and Effects.” REFACE ’03, Victoria, British Columbia, 13 Nov. 2003 <<http://swen.uwaterloo.ca/~ltahvild/Publications/REFACE03.pdf>>.
4. Fenton, Norman E., and Shari L. Pfleeger. Software Metrics: A Rigorous and Practical Approach. 2nd ed. Boston: PWS Publishing Co., 1997.
5. McCabe, Thomas J. “A Complexity Measure.” IEEE Transactions on Software Engineering 2 (1978): 308-20.
6. Yourdon, Edward, and Larry L. Constantine. Structured Design. 2nd ed. New York: Yourdon Press, 1978.
7. McConnell, Steve. Code Complete. Microsoft Press, 1993.
8. Stevens, Wayne, G. Meyers, and L. Constantine. “Structured Design.” IBM Systems Journal 13.2 (May 1974): 115-39.
9. Bieman, James M., and Linda M. Ott. “Measuring Functional Cohesion.” IEEE Transactions on Software Engineering 20 (1994): 644-57.
10. Selby, Richard W., and Victor R. Basili. “Analyzing Error-Prone System Structure.” IEEE Transactions on Software Engineering 17 (1991): 141-52.
11. Gill, Geoffrey K., and Chris F. Kemerer. “Cyclomatic Complexity Density and Maintenance Productivity.” IEEE Transactions on Software Engineering 17 (1991): 1284-88.
12. Simon, F., F. Steinbruckner, and C. Lewerentz. Metrics-Based Refactoring. Proc. of the European Conference on Software Maintenance and Reengineering, Mar. 2001.
13. Tahvildari, L., K. Kontogiannis, and J. Mylopoulos. “Quality-Driven Software Reengineering.” Journal of Systems and Software 66.3 (June 2003): 225-239.
14. Tourwe, T., and T. Mens. Identifying Refactoring Opportunities Using Logic Meta Programming. Proc. of the European Conference on Software Maintenance and Reengineering Mar. 2003.
15. Kataoka, Y., T. Imai, H. Andou, and T. Fukaya. A Quantitative Evaluation of Maintainability Enhancement By Refactoring. Proc. of the IEEE

## COMING EVENTS

### August 14-17

*CCCT Conference: Computing, Communications, and Control Technologies*  
Austin, TX  
[www.iisci.org/ccct2004](http://www.iisci.org/ccct2004)

### August 19-20

*2004 ACM-IEEE International Symposium on Empirical Software Engineering*  
Redondo Beach, CA  
[www.isese.org](http://www.isese.org)

### August 23-27

*International Conference on Practical Software Quality Techniques PSQT 2004 North*  
Minneapolis, MN  
[www.qualityconferences.com](http://www.qualityconferences.com)

### September 11-17

*20th IEEE International Conference on Software Maintenance*  
Chicago, IL  
[www.cs.iit.edu/~icsm2004](http://www.cs.iit.edu/~icsm2004)

### September 13-16

*Embedded Systems Conference*  
Boston, MA  
[www.esconline.com/boston](http://www.esconline.com/boston)

### September 20-23

*Software Development Best Practices*  
Boston, MA  
[www.sdexpo.com](http://www.sdexpo.com)

### September 24-26

*IPSI 2004 Stockholm*  
Stockholm, Sweden  
[www.internetconferences.net](http://www.internetconferences.net)

### April 18-21, 2005

*2005 Systems and Software Technology Conference*



Salt Lake City, UT  
[www.stc-online.org](http://www.stc-online.org)

- International Conference of Software Maintenance, Oct. 2002.
16. Shepperd, M., and D. Ince. "Metrics, Outlier Analysis, and the Software Design Process." *Information and Software Technology*. Mar. 1989: 91-98.
17. Weiser, M. "Program Slicing" *IEEE Transactions on Software Engineer-*

- ing* SE-10 (4) (July 1984): 352-357.
18. Verbaere, Mathieu. "Program Slicing for Refactoring." Masters Thesis. University of Oxford, Sept. 2003 <<http://web.comlab.ox.ac.uk/oucl/research/areas/progtools/projects/nate/doc/MScThesis.pdf>>.

## About the Authors



**Lt. Col Ricky Sward, Ph.D.**, U.S. Air Force, is an associate professor of Computer Science at the U.S. Air Force Academy. He is currently the deputy head for the Department of computer science and the course director for the senior-level two-semester Software Engineering capstone course. Sward has a doctorate in computer engineering from the Air Force Institute of Technology where he studied program slicing and reengineering of legacy code.

**Department of Computer Science**  
**2354 Fairchild DR**  
**STE 6GI01**  
**USAF Academy, CO 80840**  
**E-mail: ricky.sward@usafa.af.mil**



**A.T. Chamillard, Ph.D.**, is an assistant professor of Computer Science at the University of Colorado at Colorado Springs where he teaches the core Master of Engineering in software engineering courses. He also currently provides software engineering consulting services to a Department of Defense agency. Chamillard spent over six years as a project manager in the U.S. Air Force, and was also an associate professor of computer science at the U.S. Air Force Academy where he taught for six years. He has a doctorate in computer science from the University of Massachusetts, Amherst.

**Computer Science Department**  
**University of Colorado at**  
**Colorado Springs**  
**1420 Austin Bluffs PKWY**  
**Colorado Springs, CO 80933-7150**  
**E-mail: chamillard@cs.uccs.edu**



**David A. Cook, Ph.D.**, is a senior research scientist at The AEGIS Technologies Group, Inc., working as a verification, validation, and accreditation agent in the modeling and simulations area. He is currently supporting the Airborne Laser program and has more than 30 years experience in software development and management. He was formerly an associate professor of computer science at the U.S. Air Force Academy, a deputy department head of the Software Professional Development Program at the Air Force Institute of Technology, and a consultant at the U.S. Air Force Software Technology Support Center. Cook has published numerous articles on software-related topics. He has a doctorate in computer science from Texas A&M University.

**AEGIS Technologies Group, Inc.**  
**6565 Americas PKWY NE**  
**STE 975**  
**Albuquerque, NM 87110**  
**Phone: (505) 881-1003**  
**Fax: (505) 881-5003**  
**E-mail: dcook@aegistg.com**

# Right Sizing Quality Assurance

Walt Lipke  
Oklahoma City Air Logistics Center

*Generally, quality assurance (QA) functions are sized at the direction of management and are rarely sized commensurately with their need. Over the years, influenced strongly by in-vogue attitudes and real-world circumstances, the size of the QA function has exhibited extremes: (1) inordinately large after an embarrassing product failure, or an executive's overreaction from attending a W.E. Deming seminar, or (2) completely eradicated when perceived to be unneeded or too expensive. This article introduces quality efficiency indicators that facilitate right sizing the quality assurance function, i.e., sizing QA to the customer's need, or the producer organization's own quality goals. The interpretation and application of the indicators is explained, and a simple example is provided demonstrating the calculation for sizing the QA function.*

After a decade of performing process improvement, rework for our organization's software development projects was dramatically reduced from approximately 75 percent of the total effort to a very low value of 3 percent. When the percentage was high, rework was easily identified; for a small amount of quality assurance (QA) effort, a large quantity of rework was generated. As our production process improved, it became increasingly more difficult to identify defects. With rework now at 3 percent, we began to examine the economics of further improvement and the possibility of reducing the QA effort. Economically, the concept arises of right sizing the QA function with respect to the needs of the customer(s) or the quality goals of the producer organization.

## Background

Generally speaking, companies are concerned with the quality of their products. Consequently, an organizational entity exists that is devoted to performing reviews, inspections, and testing for conformity to the product requirements, i.e., the QA function. However, the QA function is a cost affecting the price of a company's products. There is a cost for quality; it is not free. Thus, the QA function is connected to economic benefit.

At a minimum, QA functions should be sized sufficiently to satisfy the customer's requirement for product quality. In conflict, several pressures influence the size of the QA function. The customer wants the product at a low price with no flaws. The producer wants to *make money*, be competitive, and increase business – QA is a cost to be trimmed. Clearly, it is impossible to simultaneously satisfy these parties.

There are conflicting dynamics within the producer's organization, too. In competitive areas (multiple producers of the same product), the marketplace decides the product price. In turn, this places a constraint on the amount of rework and quality assurance the product can have and still be competitively priced. Regardless, the QA function

has the desire to achieve zero defects for the entire production process and believes it is in the best interest of the company to support this goal. However, a defect-free product most likely will not be affordable. Without some balance to the interests of the QA function, it can become too large. These are the influences of the classic market-share dilemma.

From the producer's perspective, QA needs to be efficient and rework minimized. Minimizing the cost of QA and rework makes the product more competitively priced and maximizes profit. A good production process will satisfy nearly all of the customer's requirements without QA, i.e., quality is built in, not inspected in. Likewise, a good QA process will identify most, if not all, of the nonconformance.

The customer, reasonably, cannot expect a perfect product. However, customers can mitigate their risk of purchasing poor products by testing performance and inspecting physical details during the production process and prior to accepting delivery. By performing product acceptance, the customer increases his cost of acquiring the product. His investment in product testing and inspection is an expense, and a portion of the product price is attributable to the customer-generated rework.

Defects not identified by the producer are subject to detection by the customer during his product testing and inspection. The customer's perception of product quality is created largely from the defects he identifies. To gain repeat business or good references for new business, the producer strives to minimize the defects that propagate, or leak, through his production and QA processes.

## Quality Process Indicators

Minimizing the expenditure for QA yet meeting the customer's quality requirement is not a simple matter. To accomplish the task, management must have indicators for improving the processes and achieving the needed level of quality. In the following,

three measures of quality efficiency are proposed for determining the effectiveness and stability of the production and quality processes.

To better understand the subsequent discussion, the intended meaning of defects and rework is provided. The product requirements are the potential defects. A defect is nonconformance to a requirement, created as a function of the production process and its employees. Defects may be identified at any time during the production process up to customer acceptance. Rework results from the defects identified. Therefore, rework is a function of the QA process, QA employees, and customer testing and inspections. In mathematical form, defects and rework are expressed as follows:

**Defects = f(production process, production employees)**

**Rework = f(QA process, QA employees, customer verification)**

For an adequate understanding, a producer must have knowledge of the effectiveness of production and QA processes. Also, the producer needs to have information concerning the quality efficiency (QE) of the QA process itself. By having this information, the processes can be improved and the amount of improvement can be quantified.

Three measures are proposed to satisfy the information needed by the producer. These measures provide the capability for determining the goodness of the production and QA processes. The definitions of the measures are described below:

$$QE_1 = R(\text{process}) / R \quad 1)$$

where,

**R = total rework costs**  
**R = R(process) + R(customer)**  
**R(process) = rework from the production process**

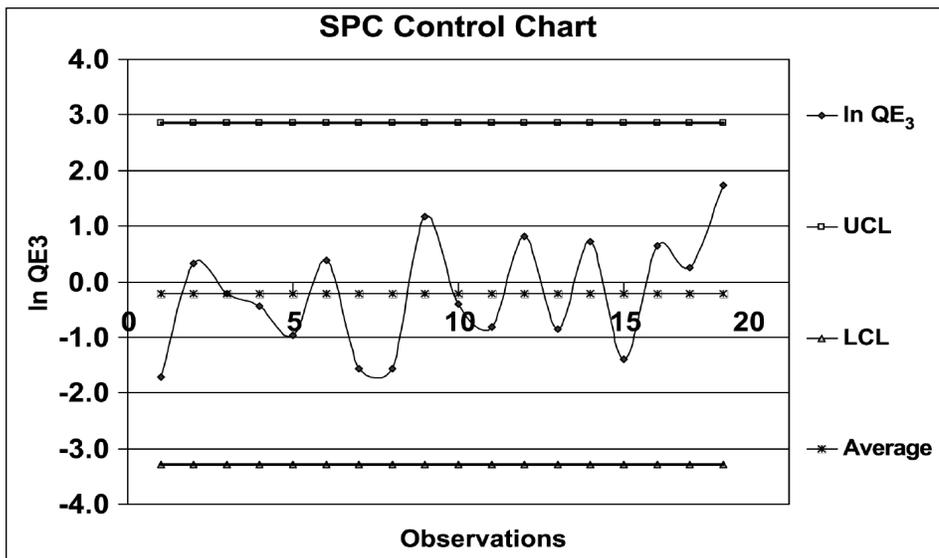


Figure 1: Statistical Process Control Chart

**R(customer) = rework from the product inspections and testing conducted by the customer**

The indicator is a measure of the QE of the quality process. When  $QE_1$  indicates the customer identifies an excessive number of defects, improvement is needed from the QA process and its employees. Rework can come from non-requirements when good requirements management is not practiced. However, only rework from nonconformances to requirements is used in the calculation of the indicator.

$$QE_2 = P / T \quad 2)$$

where,

**P = production costs**  
**T = P + R + Q = total effort**  
**Q = quality assurance costs**

The indicator is a measure of efficiency of the production process. When  $QE_2$  indicates excessive defects, the performance of the production process and its employees requires improvement.

$$QE_3 = R(\text{process}) / Q \quad 3)$$

The indicator is a measure of efficiency of the production and QA processes. When  $QE_3$  is much greater than 1.0, the production process is examined for improvement. Conversely, when  $QE_3$  is much less than 1.0, the QA process requires review and improvement.

**Analysis**

Satisfactory QA is indicated when all three indicators approach the value 1.0. As seen from examining the equations, it is possible for  $QE_1$  and  $QE_3$  to be equal to 1.0. However, it is not possible for  $QE_2$  to have a

value of 1.0 when R and Q are not zero. The only condition for which  $QE_2$  can equal 1.0 is when  $R=0.0$  and  $Q=0.0$ , i.e., perfect process quality. It has been written that the minimum value of QA needed to maintain a high achieving quality process is 2.5 percent of the total effort [1]<sup>1</sup>. Thus, the maximum value expected for  $QE_2$  is 0.975.

The indicator  $QE_1$  has the most influence on the customer's perception of product quality. Of the three indicators, it is the only one for which perfection ( $QE_1=1.0$ ) can be consistently achieved. Thus,  $R(\text{customer}) = 0.0$  (i.e., zero defects are identified by the customer) can (and should) be an expected outcome of the production and QA processes<sup>2</sup>.

Under normal conditions, the value of  $QE_3$  will approach 1.0, when the QA process is effective. However, as  $QE_1$  and  $QE_2$  approach the value of 1.0,  $QE_3$  will approach zero. Using the equation for  $QE_3$ , this circumstance is more clearly understood. As the production process improves and approaches zero defects, the numerator,  $R(\text{process})$ , approaches 0.0. Concurrently, the denominator, Q, approaches its minimum value (2.5 percent of total effort), and thus,  $QE_3$  approaches 0.0.

Indicators  $QE_1$  and  $QE_2$  may be used as evidence of defect prevention. The concept of defect prevention is that the QA process minimizes or eliminates the propagation of defects to the customer, and the production process has been optimized such that rework and QA are minimized [2].  $QE_1$  provides information concerning the amount of defect leakage from the QA process to the customer. Simultaneously,  $QE_2$  provides information concerning the optimization of the production process. Taken together, these indicators show how well defect prevention is being achieved. When  $QE_1$

approaches 1.0 and  $QE_2$ , simultaneously, nears 0.975, the production and QA processes are performing defect prevention at a level nearing perfection.

The indicators,  $QE_1$ ,  $QE_2$ , and  $QE_3$ , are to be observed as both cumulative<sup>3</sup> and periodic values. The cumulative number provides information as to the status of the process over a span of time. The periodic values yield trend information and help to answer the question, "Is the process improving, or is it getting worse?"

**Quality Function Sizing**

When the indicators  $QE_1$ ,  $QE_2$ , and  $QE_3$  are satisfactory with respect to the customer's needs or the organization's quality goals, and  $QE_3$  is in statistical control, the QA function can be reliably sized. Likewise, the QA function can be sized for a new project using the data from a historical project, as long as the production and quality processes are unchanged. A statistical process control (SPC) control chart of the periodic observations of  $QE_3$  is used to determine if the quality and rework processes are in control [3]<sup>4</sup>. The control charts may also be used as a *Rum* chart [3] for detecting the process reaction to improvements implemented.

As an example, Figure 1 is a SPC control chart created from real project data, shown in Table 1. As clearly seen in Figure 1, all observed values are within the upper and lower control limits shown as upper confidence limit (UCL) and lower confidence limit (LCL), respectively. Thus, the processes governing  $QE_3$  are statistically stable.

Upon achieving statistical control, the QA function is sized from the periodic observations of  $Q/T$ , i.e., the quality investment as a fraction of total effort. From the average of these observations and their statistical variation, a 95 percent confidence value can be calculated for  $Q/T$ . At 95 percent confidence, we are 95 percent certain the actual QA requirement will be less than the size of the function created. Sizing QA at 95 percent confidence mitigates the risk of not sizing the QA function adequately.

The 95 percent confidence we are seeking is the UCL of the 90 percent confidence interval; 10 percent of the normal distribution is outside of the confidence interval, 5 percent below the LCL, and 5 percent above the upper limit.<sup>5</sup> Having a QA requirement less than the lower confidence limit is not a concern; therefore, only the upper limit is used.

The 95 percent confidence limit,  $(Q/T)u$ , is used in a linear relationship between the total effort cost (T) and the size of the QA function, i.e.,

$$Q = (Q/T)u \times T$$

where,

### Q is the expected cost for QA

This relationship is to be used with the project plan, specifically the monthly expenditures for total effort, to *right size* the application of QA resources. Performing the computations for the monthly values of Q will yield a funding profile for the QA function. In turn, this profile may be converted and used as the staffing profile.

To compute the 95 percent confidence limit, the periodic observations of Q/T are used as logarithms to make the statistical calculations<sup>4</sup>. The standard deviation  $\sigma$  is estimated for  $\ln(Q/T)$ , while the logarithm of the cumulative value,  $(Q/T)_c$ , is the estimate for the average value. Therefore, the confidence limit is first computed as a logarithm. Thus, the equation for the calculation of the 95 percent confidence limit follows:

$$(Q/T)_u = \text{antilog} [\ln(Q/T)_c + 90\% \text{ confidence interval}] \text{ (see Note 6)}$$

The antilog value,  $(Q/T)_u$ , is the appropriate number for the sizing computation.

Using the project data from Table 1, the value of  $\ln(Q/T)_c$  is computed to equal -2.7662, with a standard deviation,  $\sigma=0.5048$ . From the values for  $z$  (=1.645),  $\sigma$ , and  $n$  (=18), the 90 percent confidence interval is calculated to be 0.1957. Adding  $\ln(Q/T)_c$  and the 90 percent confidence interval yields the value -2.5705. The value of  $(Q/T)_u$  is then computed from the antilog of the sum, and is determined to be 0.0765. For this project, the right size for the QA function is computed to be 7.65 percent of the total effort.

### Summary

To economically apply QA requires three indicators of quality efficiency converge and approach 1.0. Two indicators are measures of defect leakage to the customer and from the production process, and the third measures the efficiency of identifying defects. The indicators are useful for improving the production and QA processes. Ultimately, upon achieving *in control* processes, the quality assurance function can be sized commensurately with the customer need, or the producer's quality goals<sup>7</sup>. ♦

### References

1. Crosby, Philip B. Quality Is Free. New York: McGraw-Hill, 1979.
2. Paulk, M., et al. Capability Maturity Model for Software, Version 1.1. Software Engineering Institute, CMU/SEI-93-TR-24. Feb. 1993.
3. Pitt, H. SPC for the Rest of Us. Reading, MA: Addison-Wesley, 1995.

4. Crow, E. L., et al. Statistics Manual. New York: Dover, 1960.

### Notes

1. *High achieving* means nearly all of the producer's effort is in production. Extremely small efforts are performed for QA and rework to achieve the product requirements. In the author's opinion, very good quality for software producers would be  $QE_1 \geq 0.98$ ,  $QE_2 > 0.8$ , and  $QE_3$  between 0.6 and 1.2. World-class quality would be characterized by  $QE_1 = 1.0$ ,  $QE_2 > 0.9$ , and  $QE_3$  between 0.8 and 1.1.
2. The customer is still at risk of product defects, even when  $R(\text{customer}) = 0.0$ . Defects may be missed by the customer's inspection and testing.
3. Cumulative values for the three quality efficiency indicators are computed using the total values of the two parameters involved. For example, the cumulative for  $QE_2$  would use total values for P and T.
4. When applying statistics, it is recommended to use the logarithm values of the periodic observations of  $QE_3$  and Q/T. These parameters have been statistically tested as logarithms, and appear to be normally distributed. The results of statistics applications such as SPC and Confidence Interval are improved when the representation of the observations approximates a normal distribution.
5. The Confidence Interval is the region surrounding the computed average value within which the true value lies with a specified level of confidence. The end points of the interval are the Confidence Limits. The equation for the Confidence Limits is :

$$\langle x \rangle \pm z (\sigma/\sqrt{n})$$

where,

$\langle x \rangle$  is the average value of  $x$ , while  $z$  is from the standard unit normal distribution and corresponds to the area selected (for this application,  $z = 1.645$  at 95 percent of the distribution area),  $\sigma$  is the standard deviation of the observations of  $x$ , and  $n$  is the number of observations [4].

6. The calculation is easily performed using the capability within personal computer spreadsheet applications, such as Microsoft's Excel.
7. Sizing the QA function using the method presented in this article assumes there is a semi-smooth flow of effort, and the requirement for QA is not sporadic.

Month	Rp	Q	T
1	15	83	1784
2	234	170	2808
3	124	154	3445
4	106	165	3051
5	39	103	2303
6	546	373	6178
7	30	143	2371
8	32	154	3374
9	247	77	2020
10	53	79	2321
11	75	169	3638
12	82	36	1473
13	221	518	4294
14	227	111	1111
15	191	768	4669
16	159	84	3571
17	144	111	3218
18	449	80	2059

Table 1: *Real Project Data*

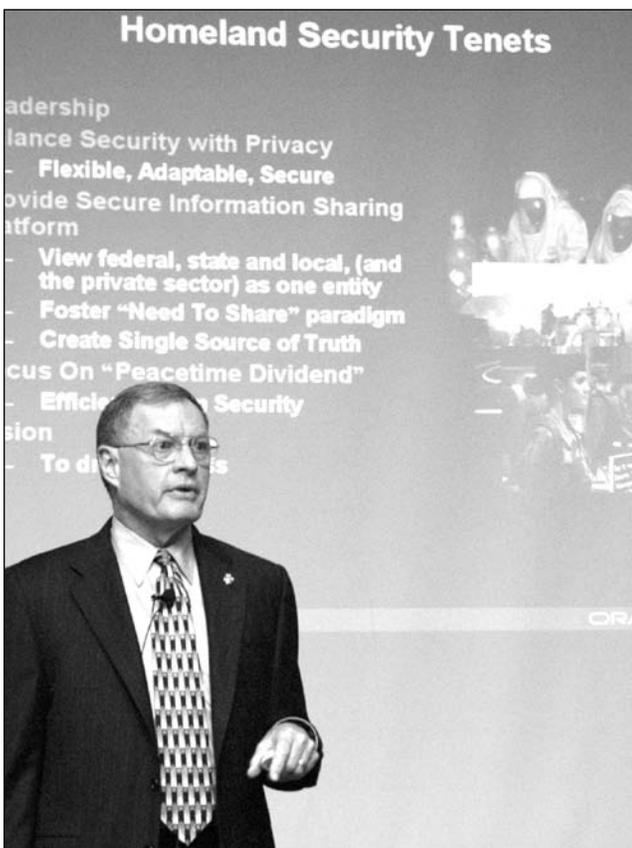
### About the Author



**Walt Lipke** is the deputy chief of the Software Division at the Oklahoma City Air Logistics Center. The division employs approximately

600 people, primarily, electronics engineers. He has 30 years of experience in the development, maintenance, and management of software for automated testing of avionics. In 1993 with his guidance, the Test Program Set and Industrial Automation (TPS and IA) functions of the division became the first Air Force activity to achieve Level 2 of the Software Engineering Institute Capability Maturity Model<sup>®</sup> for Software (SW-CMM<sup>®</sup>). In 1996, these functions became the first software activity in federal service to achieve SW-CMM Level 4 distinction. The TPS and IA functions, under his direction, became ISO 9001/TickIT<sup>®</sup> registered in 1998. These same functions were honored in 1999 with the Institute of Electrical and Electronics Engineers Computer Society Award for Software Process Achievement. Lipke is a professional engineer with a master's degree in physics.

**OC-ALC/MAS  
Software Division  
Tinker AFB, OK 73145-9144  
Phone: (405) 736-3341  
Fax: (405) 736-3345  
E-mail: walter.lipke@tinker.af.mil**



(Top photo) Participants at the 16th annual Systems and Software Technology Conference talked with exhibitors about new technologies and, in some cases, even took them for test-drives. (Photo above) retired Lt. Gen. Keith Kellogg Jr., U.S. Army, spoke at the Wednesday lunch session about current activities in homeland security.

## 16th Annual Systems and Software Technology Conference Focused on Technology to Protect America

Today's requirements to protect America have created new challenges at home and abroad. Our warfighters want rapid delivery of new capabilities, successful interoperability of complex systems, and high quality decision-support information. And they want it all now – in a timeframe that reduces days to hours, hours to minutes, and minutes to seconds. The challenge is to integrate old and new architectures, reestablish credibility of the acquisition community, renew the work force, and tackle the diverging trends in software dependency/growth and productivity.

These are the challenges that were addressed at the 2004 Systems and Software Technology Conference (SSTC) held recently at the Salt Palace Convention Center in Salt Lake City. Many of

the latest technologies and human ingenuities that make these challenges an opportunity were presented at SSTC from April 19-22 as presenters and speakers brought their expertise and experience on the conference's theme, "Technology: Protecting America," to more than 2,200 attendees from around the world.

The SSTC is one of the largest co-sponsored events for U.S. defense-related software technologies, policies, and practices.

Also this year, the conference expanded its scope and focus to include not only the software side of defense technology but also the systems engineering side as well. That is why its name has changed from the Software Technology Conference to the Systems and Software Technology Conference. However, the content of the event has not





Participants were given the opportunity to use various software applications in the computer lab during sessions throughout the week.



Maj. Gen. Kevin J. Sullivan, Commander, Ogden Air Logistic Center, (right) welcomed Maj. Gen. Conrad Ponder, U.S. Army, to the SSTC.

changed and has in fact expanded. The SSTC goal remains to provide a forum for systems and software professionals in the Department of Defense (DoD), related industries, and academia to accomplish the following:

- Learn by increasing the understanding of scientific and technical issues relevant to the mission of the DoD.
- Discover effective system and software technologies.
- Connect with attendees to exchange lessons learned in the acquisition, development, support, and management of software intensive systems.

The 16th annual SSTC featured 148 different exhibitors

and 214 speaker presentations, including the addition of two new speaker tracks. In addition to the educational and training opportunities at the SSTC 2004, the conference gave attendees a chance to network at all levels at a variety of planned events throughout the week.

The SSTC is co-sponsored by the U.S. Army, Marine Corps, Navy, Air Force, Defense Information Systems Agency, the Department of the Navy, and Utah State University Extension. The co-sponsors have already started planning SSTC 2005 scheduled for April 18-21 in Salt Lake City. ♦



The midweek entertainment break, "Swinging Through World War II," added a little festivity to the conference with dinner and a show.



Members of the Tuesday co-sponsors panel (from left) Dawn C. Meyerriecks, Defense Information Systems Agency; Diann L. McCoy, Defense Information Systems Agency; John M. Gilligan, U.S. Air Force; Maj. Gen. Conrad W. Ponder, U.S. Army; David M. Wennergren, Department of the Navy; Rear Adm. Michael A. Sharp, U.S. Navy; Brig. Gen. John R. Thomas, U.S. Marine Corps.



**Get Your Free Subscription**

Fill out and send us this form.

**OO-ALC/MASE**

**6022 FIR AVE**

**BLDG 1238**

**HILL AFB, UT 84056-5820**

**FAX: (801) 777-8069 DSN: 777-8069**

**PHONE: (801) 775-5555 DSN: 775-5555**

Or request online at [www.stsc.hill.af.mil](http://www.stsc.hill.af.mil)

NAME: \_\_\_\_\_

RANK/GRADE: \_\_\_\_\_

POSITION/TITLE: \_\_\_\_\_

ORGANIZATION: \_\_\_\_\_

ADDRESS: \_\_\_\_\_  
\_\_\_\_\_

BASE/CITY: \_\_\_\_\_

STATE: \_\_\_\_\_ ZIP: \_\_\_\_\_

PHONE: (\_\_\_\_) \_\_\_\_\_

FAX: (\_\_\_\_) \_\_\_\_\_

E-MAIL: \_\_\_\_\_

**CHECK BOX(ES) TO REQUEST BACK ISSUES:**

- APR2003  THE PEOPLE VARIABLE
- MAY2003  STRATEGIES AND TECH.
- JUNE2003  COMM. & MIL. APPS. MEET
- JULY2003  TOP 5 PROJECTS
- AUG2003  NETWORK-CENTRIC ARCHT.
- SEPT2003  DEFECT MANAGEMENT
- OCT2003  INFORMATION SHARING
- NOV2003  DEV. OF REAL-TIME SW
- DEC2003  MANAGEMENT BASICS
- MAR2004  SW PROCESS IMPROVEMENT
- APR2004  ACQUISITION
- MAY2004  TECH.: PROTECTING AMER.
- JUN2004  ASSESSMENT AND CERT.

**TO REQUEST BACK ISSUES ON TOPICS NOT LISTED ABOVE, PLEASE CONTACT KAREN RASMUSSEN AT <STSC.CUSTOMERSERVICE@HILL.AF.MIL>.**

**WEB SITES**

**AT&L Knowledge Sharing System**

<http://akss.dau.mil/jsp/default.jsp>  
The Acquisition, Technology, and Logistics (AT&L) Knowledge Sharing System (AKSS) was launched in October 2002 to replace the Defense Acquisition Deskbook. Like its predecessor, AKSS continues to provide acquisition information for all Department of Defense service components and across all functional disciplines. AKSS serves as the central point of access for all AT&L resources and information, and to communicate acquisition reform. As the primary reference tool for the defense AT&L work force, it provides a means to link together information and reference assets from various disciplines into an integrated, but decentralized, information source.

**Army Software Metrics Office**

[www.armysoftwaremetrics.org/index.asp](http://www.armysoftwaremetrics.org/index.asp)  
The Army Software Metrics Office (ASMO) provides support to U.S. Army and Department of Defense managers to develop effective software measurement programs. The ASMO supports the Army Software Test and Evaluation Panel initiative, which defines specific procedures for test and evaluation of software-intensive systems, establishes an issue-driven software metrics process to support management of Army software projects, and provides resources to better define the user and technical requirements of software-intensive systems. The ASMO provides a help desk, maintains a Web site, and develops

and disseminates computer-based training programs. The help desk provides real-time support for any inquiry during normal working hours. E-mails are welcomed also.

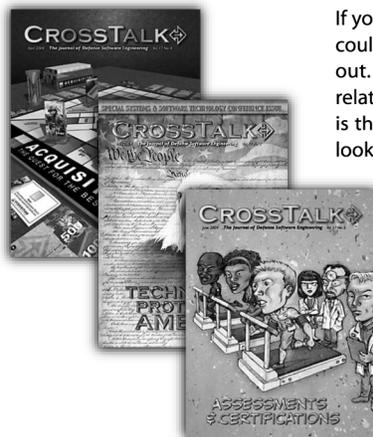
**Refactoring Home Page**

[www.refactoring.com](http://www.refactoring.com)  
This site is a simple portal for information about refactoring. There is a list of sources of information about refactoring, including various book listings, and a catalog of common refactorings, mostly taken from "Refactoring: Improving the Design of Existing Code." Many refactorings can be automated, and various tools exist to help the refactoring process. These are addressed and a mailing list is available for additional questions.

**The Quality Assurance Institute**

[www.qaiusa.com](http://www.qaiusa.com)  
The Quality Assurance Institute (QAI) is exclusively dedicated to partnering with the enterprise-wide information quality profession. QAI is an international organization consisting of member companies in search of effective methods for detection-software quality control and prevention-software quality assurance. QAI provides consulting, education services, and assessments. The QAI has recently announced the formation of a new certification program for the software project manager. The QAI Web site also features upcoming seminars and conferences, software certification, listing of local and federation chapters, on-line courses, in-house training, and news items.

**CALL FOR ARTICLES**



If your experience or research has produced information that could be useful to others, CROSSTALK can get the word out. We are specifically looking for articles on software-related topics to supplement upcoming theme issues. Below is the submittal schedule for three areas of emphasis we are looking for:

**Risk Management**

*December 2004*

Submission Deadline: July 19, 2004

**Open Systems/Open Source Software**

*January 2005*

Submission Deadline: August 16, 2004

**Personal Computing**

*February 2005*

Submission Deadline: September 20, 2004

Please follow the Author Guidelines for CROSSTALK, available on the Internet at <[www.stsc.hill.af.mil/crosstalk](http://www.stsc.hill.af.mil/crosstalk)>. We accept article submissions on all software-related topics at any time, along with Letters to the Editor and BackTalk.



# Movie Physics and the Software Industry

We go to movies for entertainment, and of course, most films are fiction, not real; they are make-believe. However, when a scene blatantly disregards the laws of physics it is enough to make an engineer retch. Yet Hollywood's bad physics can provide insight into our own industry's faux pas. For example, consider the following:

## Flashing Bullets

An action movie's climax is the gunfight complete with ricocheting bullets that emit bright flashes of light upon impact with anything on the set. While the flashing bullets are exciting, they are highly unlikely. The majority of bullets are made of copper-clad lead that does not spark when struck, even by steel objects. Professionals employ the same copper-lead-alloy in flammable work areas to prevent sparks.

In software engineering, flashy product features with marginal value are the flashing bullets of the industry. We get so excited about new ostentatious features that we lose sight of our original product intent. Focus on the target, not flamboyant red herrings.

## Endless Ammunition

Where do all those bullets come from? A Mac 10 expends a thirty-round magazine in a mere 1.8 seconds of sustained fire, an Uzi in 3.0 seconds. Nevertheless, a movie star can sustain fire during a five-minute scene without a single reload. If you are thinking bigger magazines, think again. To sustain a three-minute burst of fire, a Mac 10 gunman would have to carry 100 pounds of lead and 3,000 cartridges cases – not very agile or likely.

In software engineering, requirements are endless. Baselines are hollow, freezes futile. Customer needs pile up and weigh a project down like 100 pounds of lead. Hit your target before you aim at the next.

## Visible Laser Beams

From security systems to light-sabers, filmmakers treat us to conveniently visible laser beams. One problem, we can only see laser light when it hits a repercussive object, revealing a dot, not a beam. Any laser pointer user can confirm this fact. True, smoke, dust, or mist can reflect the laser light and create an apparent beam, but that is short lived unless you are a heavy smoker in which case you are short lived.

Software requirements, like laser beams, are only visible when they hit home with the developer. They are not guiding beams but elusive rays of light that require reverberation and discernment. In this case, it is okay to blow smoke or sprinkle dust to better perceive and focus in on customer requirements.

## Pushy Buckshot

Now move from bullets to the famous sawed-off shotgun, blasting thugs violently backwards into the nearest plate-glass window (see cut-free glass). That seems real, right? No, using conservation of momentum, you find the velocity of the thug is proportional to the ratio of the buckshot's mass to the thug's mass, which is tiny. The net effect for an average person and standard buckshot would be 0.4 miles per hour. The average human walks at 4 miles per hour so the only direction the thug is going is down, due to another force called gravity.

Software managers are looking for silver buckshot that will change the momentum of a project. Why do we continually believe that the tiny thrust of a new technology, process, or consultant will overcome the impetus of the project? Avoid sawed-off initiatives.

## Flaming Cars

Why are movie cars always bursting into flames the instant they collide with anything? If you watch closely, some explode before they hit anything as if the gas tank gets panicky and detonates at the mere thought of collision. A car explosion requires a tank rupture that spews a fine mist of gasoline vapor-air mixture of 0.8 percent to 6 percent and a source of ignition, typically found at the other end of the car – possible but improbable.

Software project leads deal with flaming gurus whom they feel they cannot afford to lose. They tiptoe around in fear that their fumes will ignite and explode the project. Coddling prima donnas is more harmful than helpful to a project. Their ignition probability is lower than perceived. They may burn but it is improbable that your project will detonate. They are replaceable. Stick with stable, reliable, and efficient project fuel.

## Cut-free Glass

A shattered window contains thousands of

incredibly sharp dagger-like edges. Little force is required for one of these daggers to lacerate flesh. However, thespians frequently crash through plate glass without a scratch. There are individuals who have accidentally fallen through windows without sustaining serious injuries. There are people who have survived shark attacks. However, in both cases the odds are meager.

The software industry has programmers that believe they can code their way through a project a week before delivery. Even if you make the deadline without a scratch, you leave a product full of lethal shards for your customers to navigate. Fortunately, most customers have developed thick calluses from other software products. Do not procrastinate – respect your client.

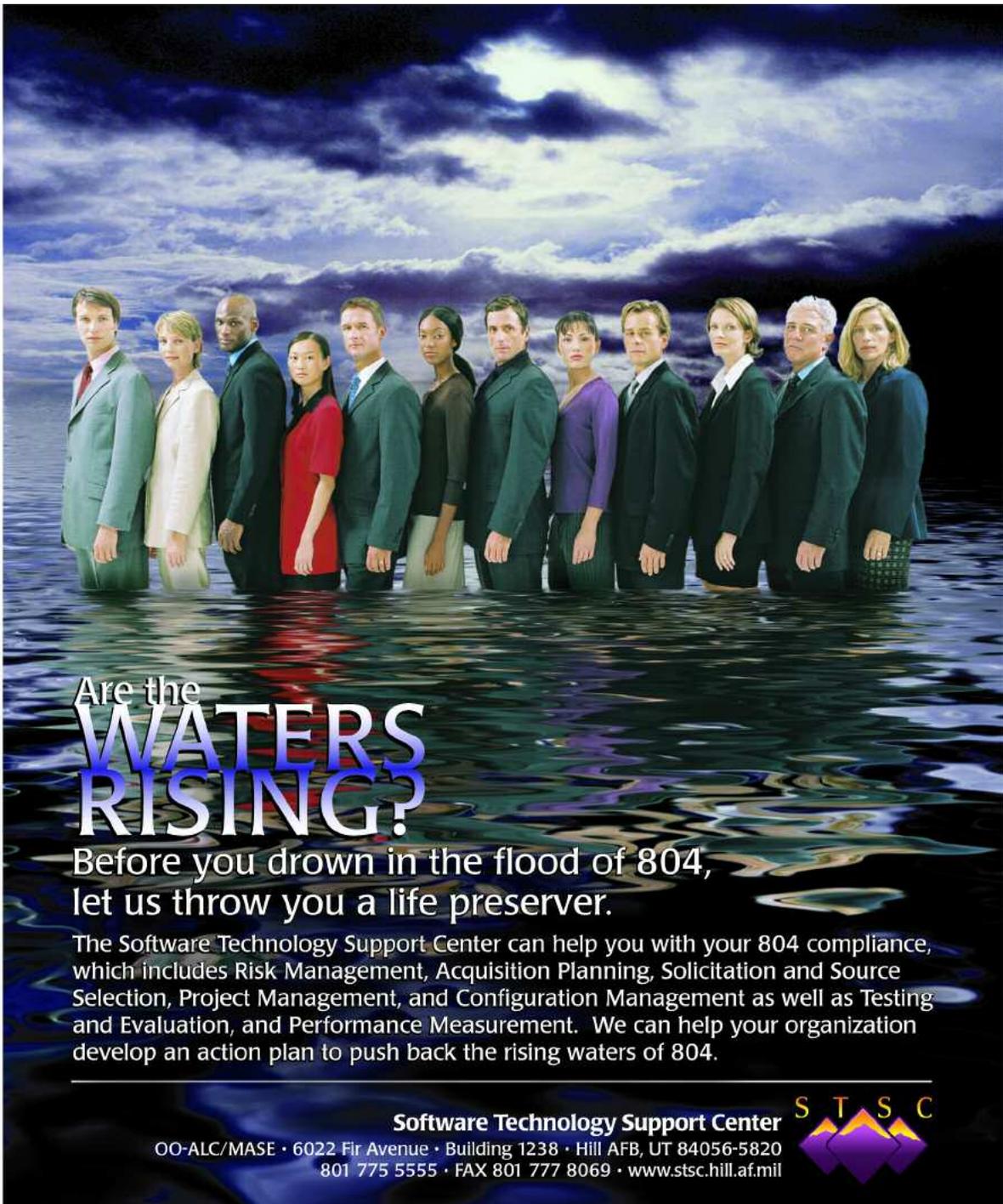
## Space Explosions

I am sorry Trekkies and Jedi, but there are two problems with explosions in outer space: With no air to transmit sound, outer-space explosions are virtually silent, save the wisp of expanding gas that passes by your X-Wing. In addition, with no gravity or air to slow them down, fragments from the explosion would travel outward until they hit something with the same kinetic energy they had during the initial blast. The first exploding Tie-Fighter would indiscriminately wipe out most of the fighters from both sides of the battle. The lucky few left would have second thoughts about setting off another shrapnel shower.

Software design errors are like space explosions. They are seldom heard and hard to spot. You do not see them coming but when they hit, they hit with more energy than they did when initiated. Once found, you spend the rest of the project dodging their debris. Spend your time and attention in the design phase. Knock out design flaws while they are small; do not wait until they are death stars.

Luckily, when we leave the theater, we realize what we saw was just a movie, fantasy, or make-believe. Unfortunately, that is not the case with software projects. These are real problems with real consequences. So sit down; grab a soda, some popcorn, and Red vines; and chew on that for a while. Enjoy your project.

— Gary Petersen  
Shim Enterprise, Inc.



# Are the WATERS RISING?

Before you drown in the flood of 804,  
let us throw you a life preserver.

The Software Technology Support Center can help you with your 804 compliance, which includes Risk Management, Acquisition Planning, Solicitation and Source Selection, Project Management, and Configuration Management as well as Testing and Evaluation, and Performance Measurement. We can help your organization develop an action plan to push back the rising waters of 804.

## Software Technology Support Center

OO-ALC/MASE • 6022 Fir Avenue • Building 1238 • Hill AFB, UT 84056-5820  
801 775 5555 • FAX 801 777 8069 • [www.stsc.hill.af.mil](http://www.stsc.hill.af.mil)



Published by the  
Software Technology  
Support Center (STSC)

### **CROSSTALK / MASE**

6022 Fir Ave.  
Bldg. 1238  
Hill AFB, UT 84056-5820

PRSR STD  
U.S. POSTAGE PAID  
Albuquerque, NM  
Permit 737