SPECIAL SOFTWARE TECHNOLOGY CONFERENCE ISSUE

CrossTalk

May 2003 The Journal of Defense Software Engineering Vol. 16 No. 5

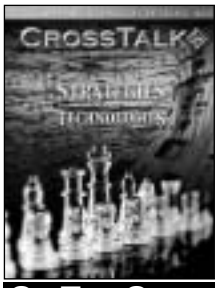
STRATEGIES AND TECHNOLOGIES

Best Practices

Software Engineering Technology

Open Forum

Departments



On the Cover
Cover Design by Kent Bingham.

CrossTalk

Sponsor	*Lt. Col. Glenn A. Palmer*
Publisher	*Tracy Stauder*
Associate Publisher	*Elizabeth Starrett*
Managing Editor	*Pamela Bowers*
Associate Editor	*Chelene Fortier*
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
CRSIP Online	www.crsip.hill.af.mil

Subscriptions: Send correspondence concerning subscriptions and changes of address to the following address. You may e-mail or use the form on p. 34.

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/xtlkguid.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: randy.schreifels@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.

Top 5 Contest Nominations Reveal Trends in COTS, E-Commerce, and Web Services

The CrossTalk staff just completed another effort to solicit and collect nominations for the U.S. Government's Top 5 Quality Software Projects. The Software Technology Support Center orchestrated a three-tier evaluation process of 70 nominations that resulted in the final selection process.

Reviewing the nominations was both motivating and educational. Collectively, the nominations demonstrated concrete examples of trends toward rapid insertion of commercial off-the-shelf technology, use of e-commerce, and application of Web services. A new step added to the evaluation process involved contacting users of the system. It was clear that users are becoming more sophisticated. Their comments concerned the architecture, hardware integration, and quality of the software – way beyond the "How did it work?" questions I was expecting. Users recognize the implications of the design on ease of upgrade and maintenance cost.

Adoption of industry standards for software development practices was evident in most nominations, particularly those developing systems with real-time requirements. Use of process asset libraries, requirements databases, product verification tracking systems, and collaborative development networks was common.

Several projects involved rapid development of applications and used end users in the design and test. The Navy's AutoREAD project, which allows flight-critical preventative maintenance tasks to be scheduled and downloaded to Pocket PCs is one example. Sailors collect readings using Pocket PCs that perform automated calculations, tolerance, and range checking to identify marginal and out-of-tolerance readings. The system was developed and fielded, and was in use aboard the carrier USS Harry Truman within four months. There were several other similar efforts that merged the use of Pocket PCs or Personal Digital Assistants as end-user data entry tools that communicated with distributed databases. At the other end of the spectrum in terms of size were several projects with millions of software lines of code and development spanning the years.

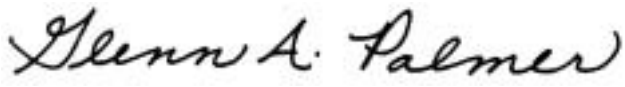
Some other techniques I found intriguing were embedding commercial data analysis and simulation software packages within mission software, automated software reengineering (e.g., ATLAS-to-C++, JOVIAL-to-C), and development of common security layers for Web services.

The nominations represent the wealth of knowledge and talent applied to Department of Defense software products, and show that we can learn by examining what went right. Look for the Top 5 issue of CrossTalk this July that will describe each of the winning projects.

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

CrossTalk is proud to announce the following winners of the 2002 U.S. Government's Top 5 Quality Software Projects. Thank you to everyone who submitted nominations; the quality of the entries was outstanding and is a tribute to the programs throughout the government. We look forward to presenting this year's awards at the 2003 Software Technology Conference.

- **Defense Civilian Pay System**
  Customer: Department of Defense
- **Enhanced Position Location Reporting System**
  Customer: U.S. Army, CECOM, PM TRCS
- **Joint Helmet Mounted Cueing System Software Upgrade**
  Customer: Joint Air Force/Navy Program Office, ASC/FBH Wright-Patterson AFB, OH
- **Kwajalein Modernization and Remoting**
  Customer: Space Missile Defense Command/ U.S. Army Kwajalein Atoll/Reagan Test Site
- **OneSAF Testbed Baseline**
  Customer: Program Executive Office – Simulation, Training, and Instrumentation



Lt. Col. Glenn A. Palmer
Director, Computer Resources Support Improvement Program

Integrated Metrics for CMMI and SW-CMM

Gary Natwick
Harris Corporation

Wednesday, 30 April 2003
Track 8: 3:50 - 4:30
Room 251 D - F

As organizations move toward the Capability Maturity Model® (CMM®) Integration℠ requiring the integration of technical and management processes across functional disciplines, the tool suites used to plan, manage, and monitor these integrated processes must also evolve to support them. One example of this is an integrated engineering metrics set to reinforce process deployment, provide effective management oversight, and ensure alignment with organizational business goals. Harris Corporation used a Goal-Question-Metric approach to develop an integrated metrics set for quantitative management of performance, progress, cost, schedule, and resources across systems, software, and hardware engineering disciplines. Institutionalization of this metrics approach resulted in achieving CMM for Software Level 4.

At the Government Communications Systems Division (GCSD) of Harris Corporation, Melbourne, Fla., integrated metrics is a key element of successful quantitative management of every program and engineering discipline. Harris Corporation achieved the Software Engineering Institute's (SEI℠) Capability Maturity Model® (CMM®) for Software (SW-CMM®) [1] Level 4 and is advancing to CMM Integration℠ (CMMI®) [2] Level 4 using integrated metrics across engineering disciplines. A SEI authorized lead appraiser performed the Level 4 SW-CMM appraisal of Harris GCSD in June 2002.

Integrated engineering metrics focus on quality, productivity, and predictability providing support data for estimating future jobs, tracking ongoing jobs, and identifying and evaluating process improvements.

Why Measure?

Harris is recognized in the industry for developing and delivering quality products; however, to advance itself in a competitive industry the company has to continually improve its overall program performance. The reason many companies in the industry are advancing their capabilities by measuring engineering processes, products, and resources is to accomplish the following:

- *Characterize* – to gain understanding of processes, products, resources, and environments, and to establish baselines for comparisons with future assessments.
- *Evaluate* – to determine status with respect to plans. Measures are indicators of when projects and processes are drifting off-track so they can be brought back under control. Evaluations also assess achievement of quality goals and the impacts of technology and process improvements on products and processes.
- *Predict* – to gain an understanding of relationships among processes and products so the values observed could be used to predict others. This is done to establish achievable goals for cost, schedule, and quality so appropriate resources can be applied. Predictive measures are also the basis for trending so estimates for cost, time, and quality can be updated based on current evidence.
- *Improve* – to identify roadblocks, root causes, inefficiencies, and other opportunities for improving product quality and process performance. Measures of current performance give us baselines to compare whether or not improvement actions are working as intended, and what the side effects may be. Good measures also help communicate goals and convey reasons for improving. This helps engage and focus the support of those working within processes to make them successful.

"One metric does not tell the whole story. You need integrated, and many times, orthogonal views of metrics to get a complete picture; trending is key."

Goal-Driven Metrics

Using the Goal-Question-Metric (GQM) [3] approach, integrated engineering metrics was derived from strategic business goals and best practices of our organization, the industry, and government. The main objective for integrated engineering metrics is to objectively measure the program health and status in relation to the following organization's goals:

- Project Management. Planning, estimating, monitoring, and controlling a project's costs, schedules, and quality.
- Process Improvement. Providing baseline data and measuring trends, tracking root causes of problems and defects, and identifying and implementing changes for process improvement.
- Organizational Vision. Effectively applying unified end-to-end engineering processes and methods encompassing proven and emerging standards/approaches for the purpose of delivering high-quality, cost-competitive system solutions to our customers.

Approach

An action team (composed of systems engineers, software engineers, program managers, and assessment experts) focused on defining the Harris GCSD's goals and ensuring the metrics needed to measure the achievement of those goals

℠ SEI, CMM Integration, and SCAMPI Lead Assessor are service marks of Carnegie Mellon University.
® Capability Maturity Model, CMM, and CMMI are registered in the U.S. Patent and Trademark Office.

were being captured at all stakeholder levels of the organization. Structured interviews were conducted with individuals representing the following four levels (from the highest to lowest) of stakeholders:

- Division management.
- Business area leadership.
- Project management and technical leadership teams.
- Functional owners of division processes.

The protocols for the interviews (individual and group) at each level of the organization were based on the results of the interviews from the previous level of the organization. Division management was asked to rate the importance of the goals in the division Strategic Guide Plan (and the goals supporting those in the plan). The business area leaders then were interviewed and asked to identify the subclass, questions, and metrics that they used, or would like to use, to achieve the goals identified by division management. The project leadership interviewees were asked to identify the questions and metrics that they used, or would like to use, to achieve the division goals and business area subgoals. The process owners were then asked to identify the questions and metrics that they would use to measure the process goals identified in the prior interviews as well as to achieve the improvement goals that the process owners identified.

The interviews were structured to correspond to the GQM [3] methodology, where issues, problems, and objectives led to the identification of measures. The interviewees were also asked to prioritize both the reasons for desiring the measurement information and the importance of the specific measures they recommended.

The Functional Analysis System Technique (FAST) [4] was used to graphically depict the linkage of each higher-level goal to lower-level goals. FAST provided a mechanism for obtaining importance ratings, by interviews, on more than 100 goals without losing the goals' context. The team analyzed the importance rating and selected the highest-level goals that spawned a set of lower-level goals with a 90 percent or greater coverage. This generated a set of top-level goals that were briefed to division management and used as the foundation for organizational metrics.

The analysis identified division goals and sub-goals: Based on the metrics currently used in the division, metrics from the industry literature, and key practices from the SW-CMM [1] and CMMI [2], metrics were identified to measure the success in achieving the goals and subgoals. GQM [3] concepts were used to validate the results of the metrics derived from the other sources and to identify any metrics that might have been overlooked. It should also be noted that several existing division metrics were dropped, as they were not directly attributable to the defined division business goals. An example of a division goal mapped to metrics using GQM follows:

- Goal: Project Management, i.e., plan, estimate, monitor, and control project quality.
- Sub-Goal: Improve customer satisfaction by reducing defects.
- Question: Where are defects introduced and removed?
- Metric: Defects detected in peer reviews and testing.

A *red team* consisting of six project teams reviewed the resulting metrics. Each project team was composed of the project's program manager, chief system engineer, chief software engineer, chief hardware engineer, and quality assurance engineers. A structured evaluation technique was used against each metric using the following criteria:

- Utility to the customer.
- Utility to the project leadership.
- Utility to division management.
- Difficulty to collect.

"Having an organizational standard tool is a must for consistency; it should be user friendly with easy access."

Results

The metrics definition effort identified metrics covering all aspects of project management and engineering performance across systems engineering, software engineering, and hardware engineering. The metrics were grouped into sets that represented a theme or view of performance familiar to each of the four levels of the organization.

The metric groupings took the form of the currently used division control panels containing up to nine metrics each (3 x 3). The control panels represented the integrated project engineering metrics distributed across systems engineering, software engineering, hardware engineering, and project engineering resources. The derived metrics differed from the pre-existing division metrics in two major areas:

- More emphasis on product quality via defect measurement and tracking.
- Additional measurement of the personnel resources' training, development, and tool support.

The metrics set supported the SW-CMM [1] and CMMI [2] Level 4 objectives of defined measurement standards. Each metric has a specified value that represents an enterprise performance goal. As data are collected, the goals are converted to control limits. The top six metrics in the example of Integrated Engineering Cost and Schedule Control Panel, shown in Figure 1 (see page 6), address the GQM as follows:

- Goal: Project Management, i.e., plan, estimate, monitor, and control project cost and schedule.
- Sub-Goal: Perform within planned cost and schedule.
- Question: How effective is the process execution versus the plan?
- Metric: Cost performance index (CPI), schedule performance index (SPI), and to-complete performance index (TCPI).

Additional information provided in the footer of these metrics is cost variance (CV), schedule variance (SV), and variance at completion (VAC).

Integrated Metrics Process

Integrated engineering metrics is used to gauge a project's progress and to alert program management of any potential risks to its quality, cost, and schedule. Each metric provides insight into systems/software/hardware engineering development products and processes and process improvement and/or organizational improvement through one of the following four major indicator categories:

- Progress. The achievement or completion of goals or commitments.
- Resources. The availability or capability of organizational assets.
- Quality. The problems and/or defects with a product or process.
- Stability. The degree of change, completeness, or effectiveness.

Everyone who uses engineering processes and/or develops engineering products utilizes engineering metrics.

Program team members are responsible for collecting and analyzing individual metrics. Project team leaders are responsible for collecting, analyzing, and reporting metrics to the program team and division management. Division management ensures the collecting and reporting of metrics, and the engineering process group conducts metrics analysis and trending. The integrated engineering metrics process has four steps: planning, collecting, analyzing, and reporting.

Planning

Planning is the first step in the integrated engineering metrics process. The collecting, analyzing, and reporting of metrics are integrated into the project plans identifying the following:

- Metrics used to support quantitative management.
- Planned and/or expected performance in the metrics, including any required goals and/or control limits.
- Variance implication and corrective action for metrics falling outside their control limits.
- Source and collection mechanism of the measurement data.
- Responsible persons for collecting measurement data, analyzing of metrics, reporting the results, and managing the engineering metrics process.

Division control limits are statistically based upon historical data. Projects use the division control limits or statistically determine their own.

Collecting

Collecting measurement data is the second and continuing step in the integrated engineering metrics process. The collection occurs at periodic intervals defined in the project plans and is monitored for completeness, integrity, and accuracy. The primary source for planning data is in the project plans. The primary source for actual data is in the accounting systems used to manage the project (e.g., financial management, configuration management, change management, and risk management) and is input into the division standard metric tool each period.

Analyzing

Analyzing metrics and making objective quantitative management decisions is the true benefit step in the integrated engineering metrics process.

Metrics are most often communicated graphically conveying a clear and easily understood message. It is better to have many graphs than it is to have many messages on one graph.

Metrics are indicators that give warnings of problems associated with issues. An issue may be tracked with several metrics that may be based on different measures. Insight into an issue typically requires statistical analysis of metrics over time and is trend-based or limit-based as follows:

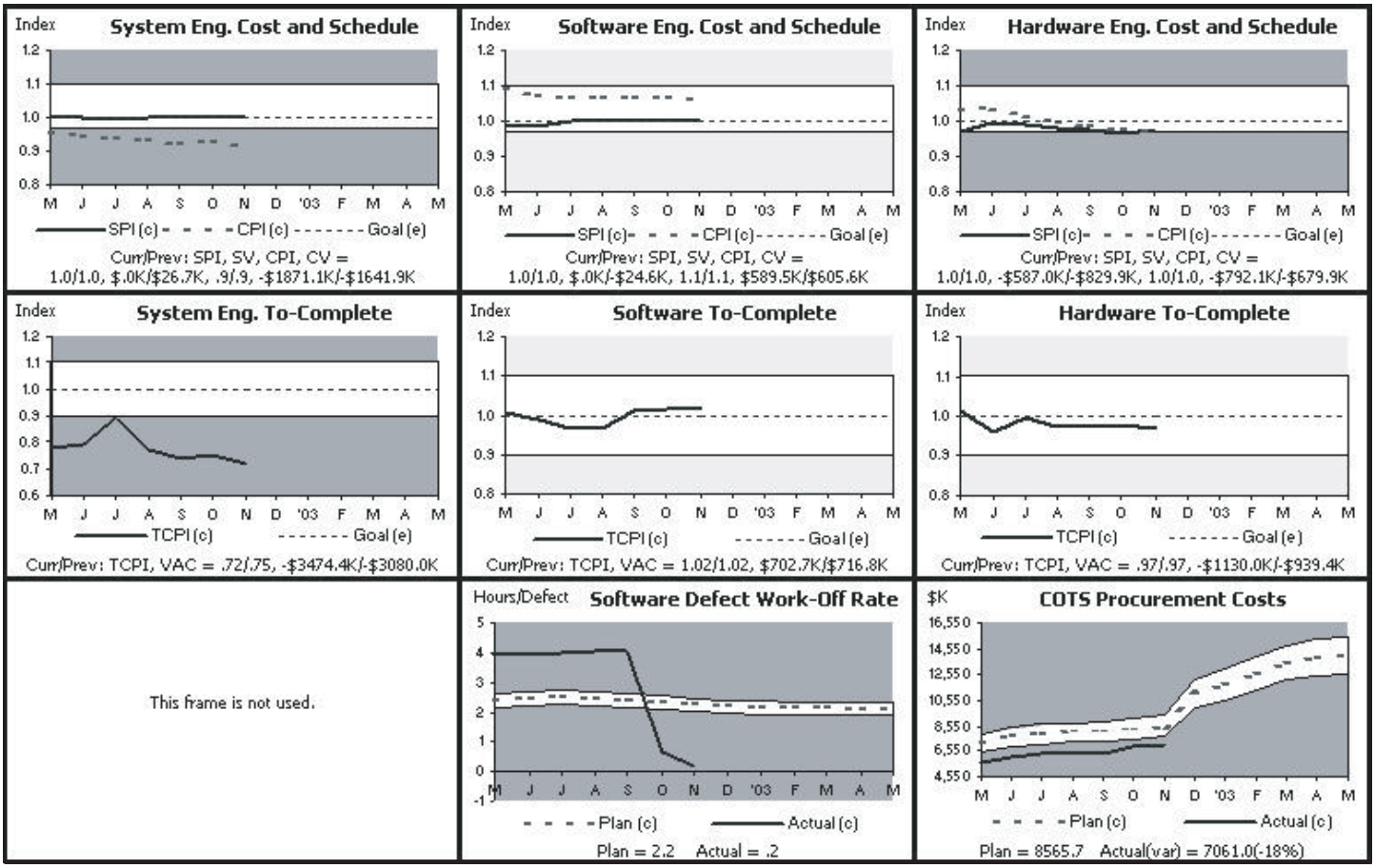
- Trend-based metrics are used when expected or planned values change regularly over time. The analysis of a trend-based metric involves determining whether the performance implied in the trend is achievable.
- Limit-based metrics are used when the expected or planned values remain relatively constant over time. The analysis of a limit-based metric requires determining whether the performance crosses its established bounds. Limits can represent norms, expected values, or constraints.

Detecting a difference, limit or trend, between planned and actual recognizes problems. If the difference exceeds the threshold of acceptable risk, then the situation is investigated and corrected.

Reporting

Reporting integrated engineering metrics is the final step in making quantitative management decisions and communicat-

Figure 1: *Integrated Engineering Cost and Schedule Control Panel*



ing to project team members, management, and customers. Reporting and reviewing metrics are integrated into the management process and occurs as soon as possible after analysis has been completed to assure that there is time for corrective action. Any metric falling outside the control limits is reviewed for variance, and corrective actions are recorded and tracked to closure. Meeting minutes are kept that record the variance explanations.

Integrated Metrics Tool

Integrated engineering metrics are collected, analyzed, and reported via the division-standard metric tool (Web client/database server) for consistency in application across the division. A required set of integrated engineering metrics is used by all projects to advance the engineering process maturity of the division.

Projects utilize additional metrics such as customer-required metrics, to complement the division-standard metric tool. A detailed definition of each engineering metric is built into the metric tool, including description, audience, purpose, method, measures, metrics, control limits, formulas, range of values, graphic information, and references. Control panels are the most common method for communicating an integrated view of engineering metric frames. A subset of the standard division metrics is presented at all program reviews.

Lessons Learned

Lessons learned from implementing a metrics program and tool within an integrated discipline work force are as follows:

- One metric does not tell the whole story. You need integrated, and many times, orthogonal views of metrics to get a complete picture; trending is key.
- Project planning is key, and data collection is the hardest.
- Having an organizational standard tool is a must for consistency; it should be user friendly with easy access.
- Cultural change is hard, so train everyone about the organizational metrics program and tool to increase acceptance and buy-in.

Conclusion

Integrated engineering metrics are required to provide effective management oversight and to ensure alignment with organizational business goals. As organizations move toward the CMMI [2] requiring the integration of technical and management processes across functional disciplines, the tool suites used to plan, manage, and monitor these integrated processes must also evolve to support them.◆

References

1. Paulk, Mark C., Charles V. Weber, Suzanne M. Garcia, Mary Beth Chrissis, and Marilyn W. Bush. Key Practices of the Capability Maturity Model®. Ver. 1.1. CMU/SEI-93-TR-25. Pittsburg, PA: Software Engineering Institute, Feb. 1993.
2. Carnegie Mellon University. CMMI℠ for Systems Engineering/Software Engineering, Ver. 1.1, Staged Representation. CMU/SEI-2002-TR-002. Pittsburg, PA: Carnegie Mellon University, Dec. 2001.
3. Park, Robert E., Wolfhart B. Goethert, and William A. Florac, Goal-Driven Software Measurement – A Guidebook. CMU/SEI-96-HB-002. Pittsburg, PA: Software Engineering Institute, Aug. 1996.
4. Kaufman, J. Jerry. Value Engineering for the Practitioner. Raleigh, NC: North Carolina State University, 1990.

About the Author

Gary Natwick is the metrics leader for the engineering process group responsible for Harris Corporation achieving the Software Engineering Institute's (SEI℠) Capability Maturity Model® for Software (SW-CMM®) Level 4 and advancing to CMM Integration℠ Level 4. Previously, he led the software engineering process group responsible for Harris Corporation achieving SW-CMM Level 3. Natwick has 30 years of software and systems engineering experience (management, development, and process improvement) with the U.S. Air Force (USAF) and Harris Corporation. He received the USAF Commendation Medal for Meritorious Service, and the Engineering Process and Golden Quill awards for advancing Harris Corporation. Natwick has a Bachelor of Science in Electrical Engineering from the University of Miami, Fla. He is a SEI Authorized Lead Appraiser (SCAMPI℠ and CBA IPI methods), and "Introduction to CMMI®" course instructor.

Harris Corporation
P.O. Box 37
Melbourne, FL 32902-0037
Phone: (321) 729-3970
E-mail: gary.natwick@harris.com

COMING EVENTS

June 2-6
Applications of Software Measurement
San Jose, CA
www.sqe.com/asm

June 7-14
Federated Computing Research Conference
San Diego, CA
www.acm.org/fcrc

June 9-13
International Conference on Practical Software Testing Techniques
Washington, DC
www.psqtconference.com

June 25-28
Agile Development Conference
Salt Lake City, UT
www.agiledevelopmentconference.com

August 25-29
QAI's Annual eXtreme Conference
Las Vegas, NV
www.qaiusa.com

September 14-19
International Function Point Users Group Annual Conference
Scottsdale, AZ
www.ifpug.org

September 22-25
AUTOTESTCON 2003
Anaheim, CA
www.autotestcon.com

October 21-24
18th International Forum on COCOMO and Software Cost Modeling
Los Angeles, CA
http://cse.usc.edu

April 19-22, 2004
Software Technology Conference 2004

Salt Lake City, UT
www.stc-online.org

Information Assurance Post 9-11: Enabling Homeland Security

David W. Carey
Oracle Corporation

Wednesday, 30 April 2003
Industry Plenary: 8:00 - 9:00
Ballroom

The demands of homeland security require information sharing on an unprecedented scale. This includes sharing information among many agencies such as foreign intelligence, domestic law enforcement, the Department of Homeland Security, the Department of Defense, federal, state, local entities, and the private sector. Most of these entities have not worked together before, and those that have will need to work even more closely together in the future. For example, a single bioterrorism incident – or the threat of one – requires that elements in agribusiness, public health, law enforcement, and foreign intelligence work seamlessly. It will take more than legislation to enable this transformation. A host of technology, policy, and cultural issues will need to be addressed. Information assurance will play a pivotal role in establishing the trust that will enable this critical transformation.

Whether the objective is thwarting terrorists beyond U.S. borders, making arrests on U.S soil, or protecting this country's critical infrastructure, the ability to share information and to do it securely is the key to an effective homeland security regime. This is a staggering undertaking.

In the new Department of Homeland Security alone, a complete information-sharing system will have to encompass some 22 current agencies, many of which are not accustomed to working with one another. These agencies must also be connected to the intelligence community, the Department of Defense (DoD), and other civilian agencies. In essence, most of the federal government needs to be involved. Now add state and local organizations from law enforcement to public health and the private sector, which – after all – controls most of our critical infrastructure.

The need for information assurance is paramount. For organizations that gather critical information – especially the law enforcement and intelligence communities – to be willing to share quality information, the computer infrastructure must be secure. This is particularly true if the information that is to be shared in a timely manner between these federal, state, and local entities and private sector organizations is actionable. If officials in these organizations do not trust that the information they provide will be protected and that it will only be shared appropriately, the information may be either withheld or sanitized to the point of uselessness.

To build this data-sharing infrastructure, a set of technical, policy, and cultural issues must be addressed. The technical issues are complex but in some ways easier to deal with than the policy and especially the cultural challenges.

Figure 1: *Data Warehousing*

Information Sharing Architectures

For some time now, the problem for law enforcement and intelligence organizations has not been a lack of information. On the contrary, authorities are often swamped with information, but it resides in separated, isolated, and discrete systems. The problem is exacerbated when the data are stored in different formats in those different systems.

Much of the time the problem is the lack of sufficient capability to establish relationships between these various bits of information. Real knowledge is found in these relationships, often more so than in the data itself. As seen in the days and weeks after 9-11, there were lots of *facts* about the individual terrorists responsible for the attacks. Because these facts were not brought together, however, no one could see the whole picture.

Of course, there are numerous ways to create the needed information-sharing system. One approach would be to create a single, national, homeland security information system that integrates data from all organizations involved. This is typical of data warehousing architectures (See Figure 1). Data are first extracted from a system. The data typically undergo some type of sanitization and transformation to normalize structure and meaning. Finally, the finished product is loaded, perhaps also combined with other system data, into a large database called the data warehouse. This process is known as extraction, transformation, and loading.

Whether or not this approach is feasible technically – and many people have suggested it is not – practical realities make it a solution that is not only unwieldy but also unlikely. Persuading all the agencies and organizations involved to defer to some central repository to protect and disseminate their information – with the concomitant loss of control that this implies if not actually entails – would be the equivalent of tilting at numerous cultural windmills in each of the agencies involved. Nor would this approach be without significant political challenges and hurdles.

Another approach would be to allow some central authority – with appropriate certificates and clearances – to reach into the myriad of databases involved and extract just that information needed for the task at hand. This distributed query approach (see Figure 2) also faces significant technical hurdles to ensure that a given query is formatted so that all the databases involved are queried in a like manner. Without that assurance, it would be impossible to have any confidence in any answer produced. Moreover, from a cultural perspective, this approach – which seemingly gives the keys to the kingdom to the enquirer – would likely be just as unpalatable as the first approach, if not more so.

The challenge then is to devise an architecture that deals with legacy systems, which probably have not been fully inventoried yet much less understood, and that does so in a way consistent with the organizational cultures involved. Although we do not have the luxury of starting with a blank piece of paper, we can make it possible for the various organizations involved to build or alter their systems in such a way that they can work in concert. These loosely integrated systems could work together to support a national strategy for homeland security. Each of the entities involved must be able to meet its organization's requirements and fit into its organization's infrastructure while adhering to standards for information sharing.

The resulting architecture would allow each organization to publish to a database that it still controls, allowing that organization to ensure that the data provided meets the requirements levied but does not contain inappropriate material (see Figure 3). Information from such individual repositories could then be amalgamated and made available to appropriate users. For example, if that newly constructed database were a terrorist watch list, then users would have an up-to-date repository of all the watch lists now extant with a pointer system to direct them to the appropriate agency for more in-depth information.

Standards

Above all, creating such an information-sharing system would require a commitment to standards. Those standards fall into three categories: data, integration, and security.

Data

When sharing data, it is not only important to establish data format standards, but also to understand the semantics of data fields and data quality, including how the data were collected. This makes it possible for organizations to compare data and establish relationships.

For example, the Department of Justice has defined a data standard called the National Incident-Based Reporting System (NIBRS). This standard defines guidelines for collecting and storing information related to a criminal incident. For example, it defines a data element for storing a person's eye color: the codes representing the allowed values for eye color, and the color corresponding to each code.

In this way, if two systems are NIBRS-compliant, the data in each can be compared easily because both use the same code to represent blue eyes, for example. Data standards like this one are critical for ensuring that once connectivity is established between systems, users will be able to compare and interpret the results.

Integration

Integration standards define how a system exposes its data and services to other systems. Web services are the emerging technology area currently being investigated for integration. Web services consist of several technologies that are standards-based. Some examples of standards for Web services include the following:

- Web Services Description Language is an Extensible Markup Language (XML) for describing Web services.
- The Universal Description, Discovery, and Integration is an open framework for describing services, discovering businesses, and integrating business services using the Internet.
- The Simple Object Access Protocol is an XML/HTTP-based protocol for accessing services, objects, and servers in a platform-independent manner.

These standards define how a system wraps up and publishes its data to other systems along with what services it provides and how to interact with these services. A system can use these standards to, in effect, say the following:

> I know all about licenses for airplane pilots in the state of Virginia. If you give me a social security number, I will check your credentials and then give you XML in the following format that includes that person's license information.

With this approach, the user does not care how the system was built, only that it can accept and answer the question. Because the federal, state, and local systems are all going to be built independently, integration standards are required for them to effectively share information.

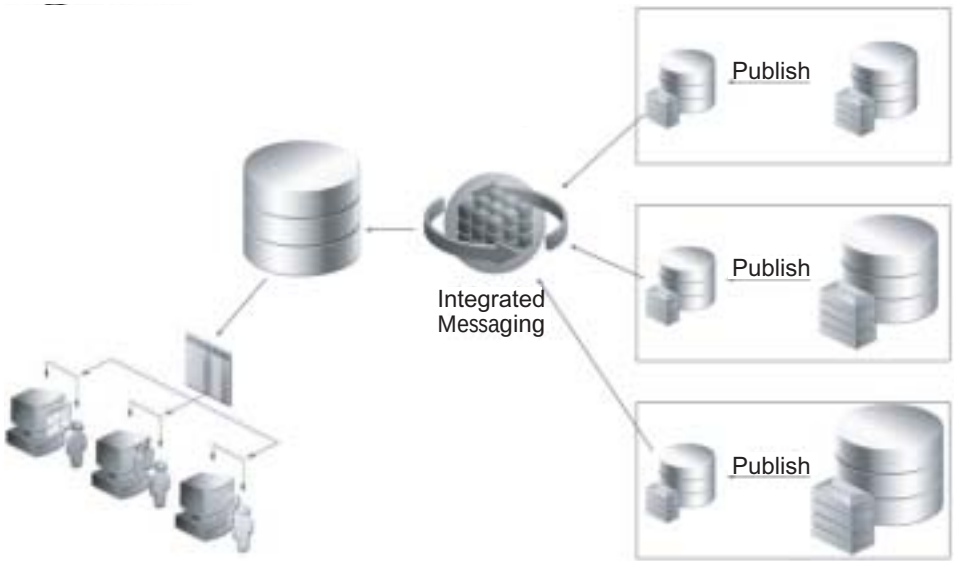
Figure 2: *Distributed Query Infrastructure*

Security

Perhaps the most important information standards are those related to security. Again, the most significant barrier to information sharing will not be the technical issue; it will be the concerns that organizations have about exposing their data to potentially insecure systems. This means that the organizations have to establish trust relationships.

To do so, at least four basic tenets of security need to be honored. First, security has to be part of the design. That is, security has to be built into the system and not bolted on afterward. When a bank is built, for example, the architects and construction personnel have thought about how people will try to break in. They will have gone through extra measures to ensure the bank's security is sound and robust. The same attention should be focused on security in information-sharing systems. Trying to secure an insecure system after it has been built simply does not work – at least elegantly.

Figure 3: *Selected Information Sharing*



Second, defense in depth is needed. By providing layers of security, assurance is given that a failure in any single defense measure does not compromise the entire system. Returning to the bank example, the architect is likely not only to use locked doors but also a time-operated bank vault, security cameras, and bulletproof glass. The bank managers are likely to add vetting procedures for bank personnel and guards.

Third, it is important to the security process that people realize risk has to be managed. While maximum security is a laudable goal, achieving that goal has to be balanced with other competing factors such as system performance, usability, administration, and even cost. A system with every possible safeguard generally suffers such performance problems that the system is not used. The workarounds devised generally sacrifice security entirely. Security is the delicate balance in risk management to secure our assets while still maintaining a usable, cost efficient, and manageable system.

Fourth, it is important to be careful about the products used. Just as there are different qualities in building materials, there are different qualities in security products. While it is largely accepted that most security is centered in the process of configuring and properly using the products, they are the foundation by which everything is built. As with most endeavors, if the foundation is weak, then it does not matter what else has been done; the system will not function appropriately.

Within the U.S. government, a level of trust is established in an operational system through the certification and accreditation process. System certification is the technical and non-technical evaluation of a system that produces the necessary evidence that is presented to the accrediting authority. The evidence needs to be comprehensive enough so that the accrediting authority can make a decision about the risk of allowing a system to be operational and to connect to other systems.

Within the government's executive and civil branches, the National Institute of Standards and Technology (NIST) is taking the lead in establishing standards and guidelines for system certification and accreditation. For the executive branch, the NIST recently released the draft publication "Guidelines for Security Certification and Accreditation of Federal Information Technology Systems."

For the DoD, the Defense Information Systems Agency (DISA) has established the "DoD Information Technology Security Certification and Accreditation Process (DITSCAP)" [1]. The DITSCAP defines both the secure design and certification process and requirements, and applies to systems within the DoD that do not include special intelligence data. Systems that process special intelligence data are certified by the Defense Intelligence Agency against the Director of Central Intelligence Directive 6/3 [2] that is similar to the DITSCAP but more stringent in its requirements.

In developing accreditable systems, it has become increasingly important to use products that have been evaluated against the security standards. This is especially true for products that support system security or provide cryptographic services. In the United States, the National Information Assurance Partnership, a collaboration between the National Security Agency and NIST, manage product evaluation. They manage the standards and independent evaluation processes required to ensure that technology providers are implementing secure products. Security products are evaluated against standards promulgated by the ISO, specifically ISO 15408, and the "Common Criteria" [3]. Encryption technology is evaluated against Federal Information Processing Standards 140-2 [4].

These evaluation standards should be enforced across this information-sharing system. To ensure that every possible step is being taken to secure data at its source, the government has taken steps to do just that for national security information systems. In July 2002, a National Information Assurance Acquisition policy went into effect for systems that contain information related to national security. National Security Telecommunications Information Security Systems Policy No. 11 [5] requires that products that have undergone independent security evaluation be used on these systems. It is imperative that policies like this one be strengthened and, more importantly, enforced through procurement policy. This policy was reiterated in DoD Directive 8500.1 [6] in October 2002.

The good news is that, in essence, the technology to build a secure information-sharing system is available today. Information can be shared widely with assurance that only the people who should see the information are granted access. Systems can be protected and users audited. And systems can be configured so that they will be available even in the event of a catastrophe. Some of the technology is already robust while other technology will become so with demand and high expectations.

In describing these secure systems, security clearances and national security classifications have not been mentioned. These protections are especially important within the intelligence community and other parts of the federal structure that are required to protect sources and methods. But it is possible to get actionable information to the people who need it without worrying about whether or not they have a national security clearance. In this case, the secure in secure information sharing means that only appropriate users are accredited to the system and that they only have the access needed to discharge their responsibilities.

Building a system that will meet these requirements is not cheap or easy, but it is doable. While reality in the information technology industry sometimes falls short of claimed performance, there are ways to ensure that does not happen. As noted, independent evaluations provide a measure of assurance that actual functionality lives up to a vendor's claims; moreover, employing evaluated products makes certification and accreditation of information technology systems easier.

It is important to buy commercial off-the-shelf products and to limit the urge to customize. Too often such customization is justified because *we are special*. This is invariably an expensive trap; the mission of homeland security is special but often the information sharing and information assurance needs are not. Point solutions or those that require 20/20 hindsight should be avoided; rather, the infrastructure to enable solutions should be created.

Other Challenges

Of all the impediments and hurdles, the technical challenges – while far from trivial – may be the easiest. As with any massive change, the principal challenge will be policy and culture.

A major policy challenge has already been met; the president has said, "Do it." Of course, myriad policy issues will have to be addressed to get it done. Setting the standards that have been discussed involves a number of key decisions. Even deciding who makes the decisions may be controversial. Who will fund and control the coordinating systems and mechanisms? Who will fund the upgrades and migration programs for the legacy systems involved, remembering as noted earlier that in the critical infrastructure community most of these entities are in the private sector? There will also be the policy decisions regarding who has access to what information.

Next are the cultural issues. At least the

technical and policy issues can be identified and worked directly. The challenge that must be faced is to build a trust relationship, frequently a challenging task when the parties know each other well. Dealing across cultural divides is often problematic in part because of the difficulty in defining the specific issues that need to be addressed.

The intelligence and law enforcement communities at the federal level work more closely together than they ever have. But there are still distinctly different cultures forged by the nature of the work. There is a common language with different definitions. For example, at the Federal Bureau of Investigation an agent is a government employee; at the Central Intelligence Agency (CIA), an agent is someone you recruit to provide intelligence or access to a foreign target; whereas in the public health arena an agent is a pathogen.

The same issues are associated with the use of acronyms. Within the CIA the National Intelligence Council – NIC – produces assessments called NIEs or National Intelligence Estimates. While within the Drug Enforcement Administration there was at one time the National Narcotics Intelligence Coordinating Committee – NNICC – which issued NIEs or Narcotics Intelligence Estimates. And, of course, CI means counterintelligence to one group in the CIA, current intelligence to another, confidential informant to most law enforcement organizations, and a computer incident within the IT community. These are purposely trivial examples, but they are symptomatic of the different mindsets forged by different missions.

These different cultures and mindsets come to the forefront when information sharing is on the table. Some of what people see as resistance to sharing is based on legal requirements. As stated earlier, by law the director of Central Intelligence is charged with protecting sources and methods. The issue of how much information can be released without revealing either source or method is legitimate. Against that backdrop though, a risk avoidance culture will ensure that less rather than more is shared. Each agency has its own security vetting process, and at present there does not appear to be a shared appreciation for what information needs to flow where.

The Task

Despite these challenges, progress can be made. Indeed, it must be made. The key is found in two old saws that fit today's challenge. The first is "think big, start small, and scale fast." Assuming that the new Department of Homeland Security will be tasked to develop this integration capability and given the appropriate authority and budget, a number of things need to happen under their auspices to start small, recognizing that in this instance small is a relative term; the task is enormous. If some other entity is given this responsibility, the tasks remain the same:

- A pilot program needs to be defined that identifies the first set of information that must be shared across organizations. This pilot must be large enough to cross organizational boundaries but small enough to allow results to be accomplished quickly. To facilitate the pilot, no more than three organizations should be involved. This also keeps to a minimum the number of security accrediting authorities that would need to be involved.
- Memorandums of understanding between the organizations need to be established along with a trust relationship at the personnel.
- Several efforts need to be started in parallel:
  - Development of system integration and data-sharing standards.
  - Design of a system architecture.
  - Definition of the security accrediting authority, process, and requirements, and the definition of the security policy and architecture.
  - Development of a prototype system to shake out the interoperability, security, and system functionality issues.

The lessons learned from this effort can then be used to *scale fast*, that is to move the prototype system to operational status and then to start a phased effort to integrate additional organizations and systems. Again, the task is huge, but it is imperative to bite off something that can work and can be done rapidly – both to get things moving and to model the culture and behavior you will need to get the bigger job done. Then scale fast.

The other old saw that is appropriate is, "the perfect is the enemy of the good." In other words, it is important to get something good started and make it better as we move along.

Both of these old saws contain an underlying note of urgency that is relevant to the task of enabling data sharing for homeland security. The formation of the new Department of Homeland Security has been likened by many to the formation of the DoD in the post-World War II era because of the enormity and otherwise daunting nature of the two undertakings. The main difference, of course, is that the DoD was formed in the aftermath of war while the new department is being formed during what unfortunately is in all likelihood the early phase of a war. The only acceptable result is success.◆

References

1. DoD Information Technology Security Certification and Accreditation Process <http://iase.disa.mil/ditscap>.
2. Director of Central Intelligence Directive 6/3 <www.fas.org/irp/offdocs/DCID_6-3_20Policy.htm>.
3. Common Criteria/ISO 15408 <www.commoncriteria.org> and <http://csrc.nist.gov/cc/index.htm>.
4. Federal Information Processing Standards 140-2 <www.itl.nist.gov/fipspubs/index.htm>.
5. National Security Telecommunications Information Security Systems Policy No. 11 <http://niap.nist.gov/ccscheme/nstissp_11.pdf>.
6. DoD Directive 8500.1 <http://iase.disa.mil> and <http://niap.nist.gov/cc-scheme/d850001p.pdf>.

About the Author

David W. Carey is vice president of Information Assurance at Oracle Corporation. He built and currently directs Oracle's Information Assurance Center, located in the company's Reston, Va., facility. The center provides a venue to demonstrate Oracle's security-related technology and for Oracle customers and partners to address a wide array of information assurance and security challenges. Before joining Oracle, Carey worked for the Central Intelligence Agency. During his 32-year career there, Carey held several senior positions, including executive director. He is a graduate of Cornell University and the University of Delaware.

Oracle Corporation
1910 Oracle Way
Reston, VA 20190
Phone: (703) 364-2126
Fax: (703) 734-1374
E-mail: dave.carey@oracle.com

Securing Your Organization's Information Assets

Dr. Bill Brykczynski and Bob Small
Software Productivity Consortium

Tuesday, 29 April 2003
Track 7: 1:00 - 1:40
Room 251 A - C

Leading companies are recognizing the criticality of their information assets and are implementing information security management systems (ISMSs) to systematically identify and protect these assets. ISO 17799 "Code of Practice for Information Security Management" is an international standard that provides a best practices framework for implementing security controls. British Standard (BS) 7799-2 "Information Security Management Systems – Specification with Guidance for Use" provides a life-cycle framework for implementing ISMSs. Accredited certification bodies have certified that more than 200 organizations worldwide meet the requirements of BS 7799-2.

All employees, whether in the public or private sector, are inextricably dependent on information in the workplace. Therefore, an organization's information assets, whether tangible or intangible, are essential. They are necessary for day-to-day productivity and for the ongoing viability of missions. Information assets are pervasive in contemporary organizations.

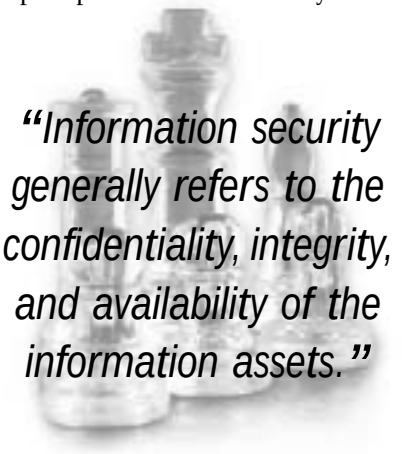
Information exists in many forms, and different types of information have different values to an organization. The impact of threats to confidentiality, integrity, and availability of information also depends on the information and an organization's mission. As information systems become increasingly interconnected, the opportunities for compromise increase.

Consider the following questions:

- Do all of your employees understand their responsibilities to protect the organization's information assets? How safe are your information assets from competitors or inadvertent exposure by your partners or customers?
- If an employee lost a laptop computer, how confident are you that you could recover from the loss with minimal disruption and compromise? What is your *risk tolerance* for such a loss?
- Are you protecting your intangible information assets, e.g., intellectual property (IP) and proprietary information, as well as your tangible information assets, e.g., computers and routers? Do your security investments complement each other to form a system of protection? Are your information assets protected in proportion to their value to the organization?
- Can your chief executive officer (CEO) tell your board of directors that your company has made a concerted effort to protect all tangible and intangible information assets?

Imagine if your company were to be sued by a customer for disclosure of sensitive financial information. The existence of an information security management system (ISMS) that was certified as meeting the requirements of a recognized standard (e.g., British Standard [(BS)] 7799-2 "Information Security Management Systems – Specification with Guidance for Use") would provide a strong defense against negligence.

Recent developments have created new tools and techniques to help organizations gain and maintain control of the complex problem of effectively securing their information assets. Before getting into the details, the following will clarify some terms:



- **Information** is what an organization has compiled or its employees know. It can be stored and communicated, and it might include customer information, proprietary information, and/or protected (e.g., by copyright, trademark, or patent) and unprotected (e.g., business intelligence) IP.
- **Information assets** are intangible information and any tangible form of its representation, including printed copies, computer files, and databases.
- **Information security** generally refers to the confidentiality, integrity, and availability of the information assets.
- **Information security management** includes the controls used to achieve information security and is accomplished by implementing a suitable set of controls, which could be policies, practices, procedures, organizational structures, and software functions.
- **ISMS** is the life-cycle approach to implementing, maintaining, and improving the interrelated set of policies, controls, and procedures that ensure the security of an organization's information assets in a manner appropriate for its strategic objectives.

ISO 17799 and BS 7799-2

In 2000, the ISO adopted the British Standard (BS) 7799-1 (Part 1) as ISO 17799, "Code of Practice for Information Security Management" [1]. ISO 17799 is a best practices framework for information security management. The standard is structured in the following 10 sections (i.e., control areas):

1. Security policy.
2. Organizational security.
3. Asset classification.
4. Personnel security.
5. Physical and environmental security.
6. Communications and operations management.
7. Access control.
8. Systems development and maintenance.
9. Business continuity planning.
10. Compliance.

Within these control areas are 36 control objectives and 127 controls. The control objectives provide guidance to the ISMS implementers on the standard's intention in a particular section. The controls are enablers; they are succinct specifications of best practices that might be incorporated into the ISMS. The controls

do not explain how they should be implemented – simply that they might be required. The standard is explicit that these controls might not be sufficient for all situations. Organizations that implement an ISMS in accordance with the standard are encouraged to create their own controls to provide additional guidance as necessary.

The following illustrates a *control objective* (from Section 3.1, Information Security Policy Document):

> Objective: To provide management direction and support for information security. Management should set a clear policy direction and demonstrate support for, and commitment to, information security through the issue and maintenance of an information security policy across the organization. [1]

The following is an example of an *enabling control* for the previous control objective (from Section 3.1.1, Information Security Policy Document): A policy document should be approved by management, published, and communicated, as appropriate, to all employees [1].

Section 3.1.1 also provides guidance on what should be included in the policy document, and how it should be communicated to the users.

Whereas ISO 17799 provides *guidance* for information security controls, BS 7799-2 states the following:

> …specifies the *requirements* for establishing, implementing, operating, monitoring, reviewing, maintaining, and improving a documented ISMS within the context of the organization's overall business risks [emphasis added]. [2]

The succinct requirements found in BS 7799-2 provide the basis for independent, third-party certification. BS 7799-2 also requires a life-cycle support system for the ISMS and recommends using the Plan-Do-Check-Act (PDCA) model. Figure 1 shows the PDCA model and illustrates the activities that occur in each phase.

To summarize, ISO 17799 is a framework (not a specification) that provides best practices for information security management. To actualize the appropriate set of controls requires the implementation of an ISMS. The BS 7799-2 specifies how an ISMS is developed and maintained in order to make operational the controls in ISO 17799.

Figure 1: *Principal Activities in the Plan-Do-Check-Act Cycle*

Accreditation, Certification, and Compliance

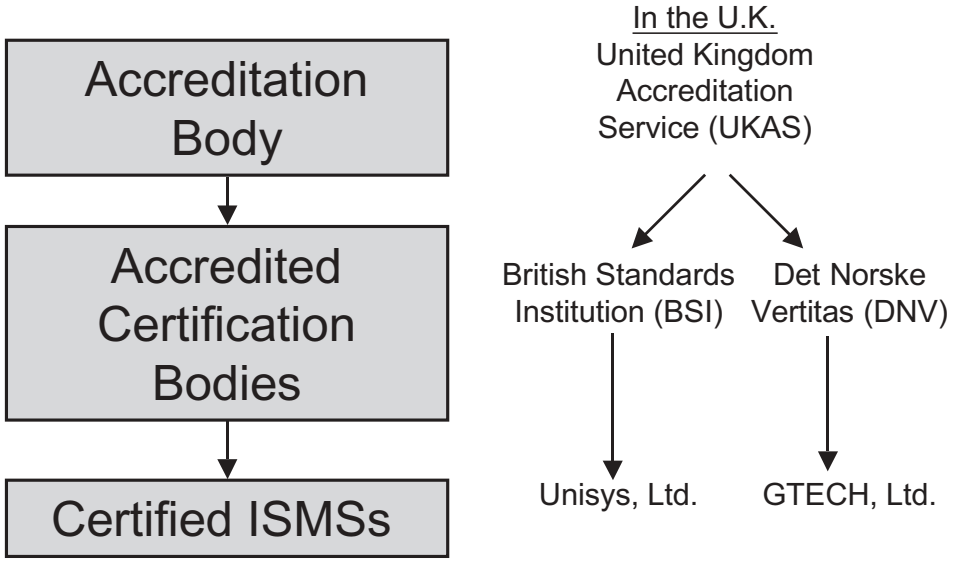
The terms accreditation, certification, and compliance are used in a variety of contexts, even within the broad area of information security. Therefore, it is important to briefly present what these terms mean in the context of ISMS:

- *Accreditation* is the means by which an authorized organization (i.e., the accreditation body) formally recognizes the competence of a certification body to assess and certify/register the ISMS of an organization with respect to published ISMS standards (e.g., BS 7799-2).
- *Certification* (or registration) is conferred by an accredited certification body on an organization upon the successful completion of an independent audit, attesting that the management system meets the requirements of a particular standard (e.g., BS 7799-2).
- *Compliance* is a self-assessment performed by an organization to validate that it has implemented a system in accordance with a standard.

Figure 2 illustrates the hierarchy of these types of organizations and credentials.

In the United States, the Registrar Accreditation Board (RAB) is the predominant accreditation body for quality management systems (ISO 9001) and environmental management systems (ISO 14000) [3]. The RAB accredits certification bodies to issue certificates to organizations fol-

Figure 2: *The Hierarchy of Credentials Leading to Certification*



lowing successful audits within these two broad areas.

In the United States today, there are no domestic accreditation or certification bodies for BS 7799-2. In the United Kingdom, the United Kingdom Accreditation Service offers accreditation to certification bodies for BS 7799-2 and other management systems standards.

For an organization considering the implementation of an ISMS, compliance is obviously a weaker claim than certification, although it is a necessary first step. For organizations that wish to implement the information security best practices in ISO 17799 today, BS 7799-2 is the companion specification for deploying and managing an ISMS. ISO has begun the formal study period on ISMSs (e.g., BS 7799-2) that will presumably lead to an ISO standard.

To date, more than 200 organizations worldwide have been certified using BS 7799-2 [4]. The Software Productivity Consortium sees increasing interest among its members in pursuing certification or at least compliance. The following three subsidiaries or divisions of consortium members have publicly acknowledged BS 7799-2 certificates:

- Fuji-Xerox in Japan.
- GTECH in Ireland and the United Kingdom.
- Unisys, Ltd. in the United Kingdom.

Unlike certification, achieving compliance is based on a self-assessment and is not a rigorously controlled credential. An organization might declare itself ISO 17799 compliant if it has implemented the information security best practices in this standard via some process. ISO 17799 is not concerned with how it is implemented and does not use the term ISMS.

Achieving BS 7799-2 certification requires achieving compliance as a first step. The PDCA model must complete at least one cycle to produce the records from the ISMS and give management the opportunity to monitor, review, and improve the system. Typically an ISMS must be in operation for at least three months before it has produced the artifacts required for audit.

An organization wishing to achieve certification for its ISMS must contract with a certification body that will assign a lead auditor to the job. As in other management systems, the size of the audit team will depend on the size and complexity of the system being audited, as well as any special domain knowledge requirements.

The audit consists of a documentation review in which the auditors review the objectives of the ISMS, and an implementation audit in which the auditors sample the artifacts of the system throughout the PDCA life cycle. The audit includes interviewing stakeholders to validate that they understand their roles and responsibilities within the ISMS.

The lead auditor will debrief the organization at the conclusion of the implementation audit and share the audit team's recommendation for certification. The certification body, in almost all cases, acts in accordance with the lead auditor's recommendation.

Critical Success Factors for Implementation

A successful ISMS, like most systems, depends on the careful balance and interaction between people, process, and technology. The principal result is to reduce risk to the organization from the potential compromise of any information asset.

Three critical success factors must be taken into account to achieve a successful ISMS implementation: effective information security, responsive management system, and organizational change management.

"In the final analysis, if the ISMS implementation does not change people's behaviors within the organization, it becomes shelfware."

In theory, risk is equal to the product of the probability of a compromising event and the impact of the compromise. Note that if either the probability is high or the impact is high, then the risk might be high. In practice, however, it is difficult to know the likelihood of most compromises, and most intangible assets do not have a monetary value. Therefore, risk analysis is not a rote exercise.

The point of implementing the ISMS is to reduce risk to the organization by improving the security of its information assets. The requisite risk analysis and mitigation planning require sound judgment to provide a solution that effectively protects the information assets from compromise to their confidentiality, integrity, and availability. In addition, confidentiality and availability may not be equally critical from organization to organization, but integrity is generally considered to be essential – without integrity, what value are the information assets?

Like other management systems, the ISMS must be designed to meet certain functional requirements, have clearly defined stakeholders with roles and responsibilities clearly defined, and include appropriate mechanisms for collecting feedback and using it to improve the system. Management systems require a commitment of time and resources.

In the final analysis, if the ISMS implementation does not change people's behaviors within the organization, it becomes shelfware. In every organization, people are already busy and do not need another set of things to do in addition to their full-time jobs. The ISMS must be intrinsic to each person's job and not merely be something else to do. Each must understand what is expected.

Peter Senge wrote the following about organizational change:

> If I stand back a distance and ask, "What's the score?" how much learning has actually been accomplished? I have to conclude that organizational inertia is winning by a large margin … Of course, there have been enough exceptions to indicate that learning is possible. However, there are many more organizations that haven't gotten to first base when it comes to real learning, and many others that have given up trying. [5]

Business Case for Implementing an ISMS

So why would an organization choose to make the investment in an ISMS and have it certified? Several questions were posed at the beginning of this article to illustrate merely a few important information security questions that might be difficult to answer. Most organizations do, in fact, have many information security controls in place. In the absence of a formal ISMS, however, these tend to be independent point-solutions rather than a unified approach to a pervasive organizational problem.

The problem is complex due to the following factors:

- The enormous quantity of information assets in most organizations.
- Assets' inherent vulnerabilities and the potential threats to their confidentiality, integrity, and availability.

- The many requirements for information security, including legal and regulatory, marketplace requirements from customers and partners, and corporate governance.

There are several reasons why an organization should make the investment in obtaining certification of an ISMS; these are outlined in the following sections.

Comprehending the Problem

In today's environment, effectively managing risk is a significant undertaking. The old adage about a chain being as strong as its weakest link certainly applies here. There are too many information assets and interfaces to be managed effectively without an ISMS. Defense in depth is an essential strategy; ultimately, however, there is no such thing as absolute security. First, understand which information assets are most important, how they might be compromised, and what the impact of a loss might be. Then, determine risk tolerance and put appropriate controls in place to ensure their security.

Earning and maintaining customer trust and protecting your organization's IP and the IP that it licenses are essential objectives in information security management. A company's reputation can easily be tarnished if management fails to take due-diligent steps to prevent compromise.

A Concrete Example

Identity theft is the fastest growing crime in the United States [6]. The largest identity-theft ring to date was broken in November 2002. It was reported that the principal suspect in the crime was a former employee of a New York company that provided software to banks and other financial institutions. The employee left the firm in March 2000 after less than a year of employment but continued to have access to private data through his own computer. This was how he was able to steal private information for criminal use [7].

What would the CEO of this company say about the need for a systematic ISMS? What would it be worth to this company to have been spared any association with this crime?

Had this company implemented an ISO 17799-compliant or BS 7799-certified ISMS, it is very likely that they would have identified many (or all) of the *links* in the chain of events that led to this disastrous outcome. They would have established controls and procedures to create a *defense in depth* that would have served them very well.

No one likes bad news, but information security risks are a part of life, and an ISMS is not a guarantee that bad things will not happen. An effective ISMS can go a long way toward reducing risk by identifying the most important assets, building appropriate risk-management solutions, and changing the culture of the organization so that information security management is intrinsic to everyone's job. In the worst outcomes, the organization is best prepared to deal with bad news, should it arise.

Organizations that have achieved certification that their ISMS meets the requirements of BS 7799-2 have objective, third-party evidence that they have taken due-diligent actions to protect the security of their information assets. At the end of the day, that is the best that can be done.

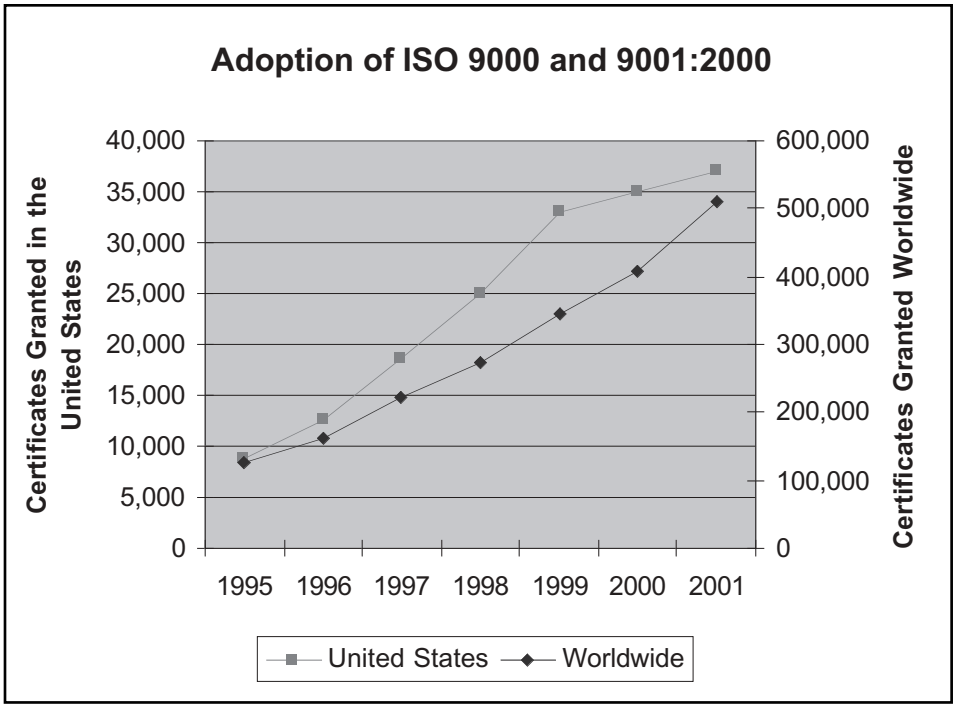


Figure 3: *Adoption of ISO 9000 as a Bellwether for ISO 17799*

Outlook for ISO 17799

This technology is still in the early adopter phase. If ISO adopts an auditing specification for 17799, the authors believe that will significantly increase demand for certification in the United States.

ISO 9000/9001 began life as a British standard. In 1995, there were almost 9,000 certificates in the United States. This grew to more than 37,000 by the end of 2001. The U.S. marketplace has hovered at around 7 percent of the global market [8]. Figure 3 illustrates the growth in the United States and worldwide markets. This might be a bellwether for ISO 17799.

Even before ISO might adopt an auditing specification for 17799, the consortium sees interest by several of its member companies to achieve BS 7799-2 certification as a discriminator in the marketplace. Others are working toward ISO compliance now as a first step toward certification in the future.

Acknowledgements

This article is based in part upon work sponsored by the Defense Advanced Research Projects Agency under Grant MDA972-01-1-0006. This article also benefited from technical review by Doron Becker, Justus Riek, Geetha Elengical, and Bernard Eydt.◆

References

1. ISO. Information Technology – Code of Practice for Information Security Management. ISO/IEC 17799:2000. Geneva, 2000.
2. British Standards Institution. Information Security Management Systems – Specification With Guidance for Use. BS 7799-2:2002. London, 2002.
3. Registrar Accreditation Board. About RAB. 2002 <www.rabnet.com/ab_main.shtml>.
4. One 7799 World, Ltd. International Register of BS 7799 Accredited Certificates <www.xisec.com>.
5. Senge, Peter, Art Kleiner, Charlotte Roberts, Rick Ross, George Roth, and Bryan Smith. The 5th Discipline: Dance of Change. New York: Doubleday/Currency, 1999.
6. Identity Theft Resource Center. Facts and Statistics: Find Out More About the Nation's Fastest Growing Crime. May 2002 <www.idtheftcenter.org/html/facts_and_statistics.htm>.
7. Weiser, Benjamin. "Identity Ring Said

to Victimize 30,000." New York Times 26 Nov. 2002.
8. ISO. Adoption of ISO 9000 and 9001:2000. <www.iso.ch/iso/en/prods-services/otherpubs/pdf/survey11thcycle.pdf>.

Additional Reading

1. For an overview of how accreditation bodies achieve their credentials, see ISO/IEC 17011, One Standard for Accreditation of All Conformity Assessment: Challenges for ISO/CASCO Working Group 18 <www.iso.ch/iso/en/commcentre/pdf/Casco0102.pdf>.
2. The ISO 17799 and BS 7799-2 standards may be obtained from <www.ceem.com/infosecurity_standards.asp>, which also contains some case studies and Web seminars.

Bob Small will also be speaking at STC 2003 on "Reducing Internet-Based Intrusions" on Wednesday, 30 April, Track 7, Room 251 A-C, from 1:50-2:30 p.m.

About the Authors

Bill Brykczynski, Ph.D., leads the security program at the Software Productivity Consortium. His professional and research interests include applied security research and software inspection processes. He has a Bachelor of Science in systems science from the University of West Florida, a Master of Science in information management from George Washington University, and a doctorate in information technology from George Mason University.

Software Productivity Consortium
2214 Rock Hill Road
Herndon, VA 20170-4227
Phone: (703) 742-7134
Fax: (703) 742-7350
E-mail: bryk@software.org

Bob Small is a principal member of the technical staff at the Software Productivity Consortium. His areas of interest include applied security research and building communities of practice. He has a Bachelor of Science in economics from Ursinus College and a Master of Science in computer science from the University of New Haven.

Software Productivity Consortium
2214 Rock Hill Road
Herndon, VA 20170-4227
Phone: (973) 984-8213
Fax: (703) 742-7350
E-mail: small@software.org

WEB SITES

Software Metrics Sites

http://user.cs.tu-berlin.de/~fetcke/measurement

The Software Metrics Sites is a guide to Internet resources on software measurement, process improvement, and related areas. Topics include electronic papers, bibliographies, and conferences on software measurement, object-oriented metrics, functional size measurement, and software process improvement. Several mailing lists that are used for discussions and the exchange of ideas can be found as well as software measurement tools that are available for downloading.

USC Center for Software Engineering

http://sunset.usc.edu/index.html

The University of Southern California's Center for Software Engineering was founded in June of 1993 by Dr. Barry W. Boehm to provide an environment for research and teaching in the areas of large-scale software design and development processes, generic and domain specific software architectures, software engineering tools and environments, cooperative system design, and the economics of software engineering. A main goal of the center is to perform research and development of practical software technologies that can reduce cost, customize designs, and improve design quality by doing concurrent software and systems engineering.

Software Technology Conference

www.stc-online.org

In its 15th year, the Software Technology Conference (STC) is the premier software technology conference in the Department of Defense and is co-sponsored by the United States Army, United States Marine Corps, United States Navy, United States Air Force, Defense Information Systems Agency, and Utah State University Extension. The STC planners anticipate more than 2,500 participants this year April 28 – May 1 in Salt Lake City from the military services, government agencies, defense contractors, industry, and academia.

MITRE

www.mitre.org

MITRE is a not-for-profit national resource that provides systems engineering, research and development, and information technology support to the government. It operates federally funded research and development centers for the DoD, the FAA, and the IRS, with principal locations in Bedford, Mass., and northern Virginia. MITRE publishes numerous periodicals and newsletters, maintains a news center, hosts numerous technology information centers, and more.

Institute for Software Research

www.isr.uci.edu

The Institute for Software Research's (ISR) mission is to advance software and information technology through research partnerships. The ISR supports research projects, develops technology, and sponsors technical and professional meetings. It works with established companies, start-ups, government agencies, and standards bodies to develop and transition the technologies to widespread and practical application. ISR was established July 1, 1999 from the Irvine Research Unit in Software.

Improving Processes for Commercial Off-the-Shelf-Based Systems

Dr. Barbara Tyson, Cecilia Albert, and Lisa Brownsword
Software Engineering Institute

Tuesday, 29 April 2003
Track 5: 1:50 - 2:30
Room 150 D - G

Organizations using commercial off-the-shelf (COTS) products in critical business and mission systems find that the traditional process of defining requirements, formulating an architecture, and then trying to find COTS products to meet the specified requirements within the defined architecture rarely works. Instead, new processes, skills, and roles are required. Many organizations that have derived substantial benefits through process improvement using capability maturity models want to know, "How should the Capability Maturity Model® Integration℠ (CMMI®) be interpreted for organizations building, fielding, and supporting a COTS-based system?" This article answers that question and describes the Evolutionary Process for Integrating COTS based systems℠, which was designed to show how COTS aspects could be addressed; it also identifies high-level guidance to facilitate the definition of appropriate work processes for developers and maintainers of COTS-based systems using the CMMI.

For many programs, commercial off-the-shelf (COTS) products offer the promise of rapid delivery to end users, shared development costs with other customers, and an opportunity to expand business and mission capabilities and performance as improvements are made in the marketplace. But the promise of COTS products is often not realized in practice. Why? An important factor is that organizations tend either to assume that COTS products can be simply thrown together, or they fall back on familiar, traditional development skills, which have been shown not to work in developing and maintaining a COTS-based system [1].

Practical experience shows that building systems using COTS products requires new skills, knowledge, and abilities; changed roles and responsibilities; and different processes [1]. Many organizations find that COTS-based systems can be complex and are often costly to build and maintain. Moreover, practitioners are finding that management and engineering processes for a COTS-based system[1] must be more (not less) disciplined.

This article characterizes the unique aspects of defining, building, fielding, and supporting a COTS-based system; describes the Evolutionary Process for Integrating COTS based systems℠ (EPIC℠) (which was designed to show how COTS aspects can be addressed [2]), and identifies high-level guidance to facilitate the definition of appropriate work processes for both developers and maintainers of COTS-based systems using the Capability Maturity Model® Integration℠ (CMMI®) [3].

℠ EPIC and Evolutionary Process for Integrating COTS based systems are service marks of Carnegie Mellon University.

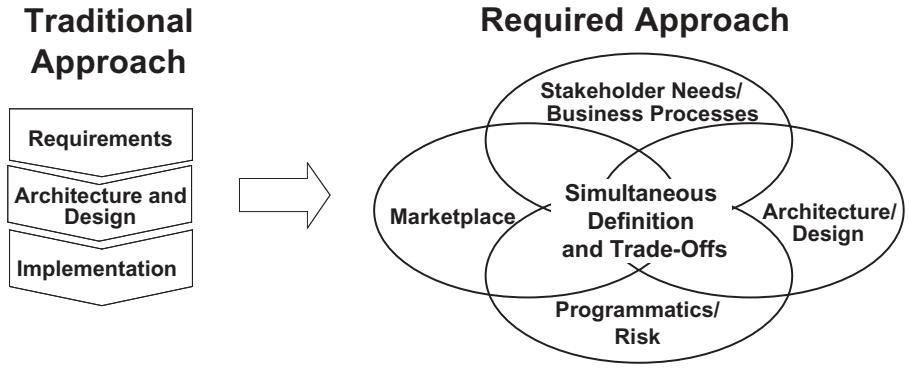
Demands of COTS-Based Systems

In custom development, a system can be *created* to meet the demands of a particular operating environment. A COTS-based system, for the most part, is *composed* of products that exist *off-the-shelf*. COTS products introduce unique dynamics and constraints that must be accommodated by any set of work processes that build, field, and support COTS-based systems such as the following:

- COTS products are developed and enhanced in response to the *vendor's perception of the needs* of a broad set of customers – the commercial marketplace – not a particular customer's needs.
- COTS products include implicit *assumptions about the way the product will be used*, which seldom match the predefined operational processes of the project's end users.
- COTS products include implicit *assumptions about the way the product will interact with other products* and the enterprise infrastructure, including dependencies on specific versions of other COTS products.
- The vendor often provides *limited visibility* into the assumptions, design, quality, and behavior of the COTS products.
- COTS products often *behave in unpredictable ways* when used in combination.
- The vendor controls the *frequency and content of COTS product releases.* In a competitive market segment, COTS products may add and/or delete functionality frequently.
- The vendor maintains the COTS product, retains data rights to the source code, and intends for the products to be *used without product modification*[2].

Despite these differences, many organizations have tried to use the more traditional approach as shown on the left in Figure 1. This approach defines the requirements, forms an architecture to meet them, and then searches the commercial marketplace for COTS products that fit into that architecture. This approach is rarely successful. COTS products do not *fit*. Organizations either resort to custom development or try to make COTS products fit by modifications. Either

Figure 1: *Fundamental Engineering, Management, and Business Change*



Note: Adapted from COTS-Based Systems for Program Managers [4]

way, they incur significant cost and schedule impacts that are repeated with each product upgrade.

In contrast, the Software Engineering Institute's experience in examining more than 50 projects attempting to build COTS-based systems shows a fundamental change is required, as shown on the right in Figure 1. To effectively leverage COTS products, knowledge of how the products behave in the operational context and a projection of how that behavior is likely to change over time must influence the definition of the solution's requirements and end-user business processes, and will drive the definition and implementation of the resulting solution. Successful projects emphasize a balance among the following four competing spheres of influence throughout the project's life:

- **Stakeholder[3] Needs and Business Processes.** An understanding of the relationship between project success and the organization's business drivers. How end-users will use the system with emphasis on the implications for end-user business processes. What the stakeholders want with emphasis on the minimum number of *must have* requirements.
- **Marketplace.** An awareness of marketplace drivers that are likely to affect the COTS products over the system's life. Knowledge of current and emerging COTS products, technologies, and standards relevant to the project.
- **Architecture and Design.** The essential elements of the system, any other systems or infrastructure with which it interacts, and the relationships among them (including structure, behavior, usage, etc.) so the components can work and evolve together.
- **Programmatics and Risk.** An understanding of the management aspects of the project, including the impact of implementing any needed changes to the end users' operational processes. What the project and end-user community can tolerate in terms of cost, schedule, and risk.

While three of the spheres from the required approach in Figure 1 have analogues in traditional development processes, the marketplace is a potent addition. To accommodate the marketplace, each sphere must be defined based on knowledge of the marketplace.

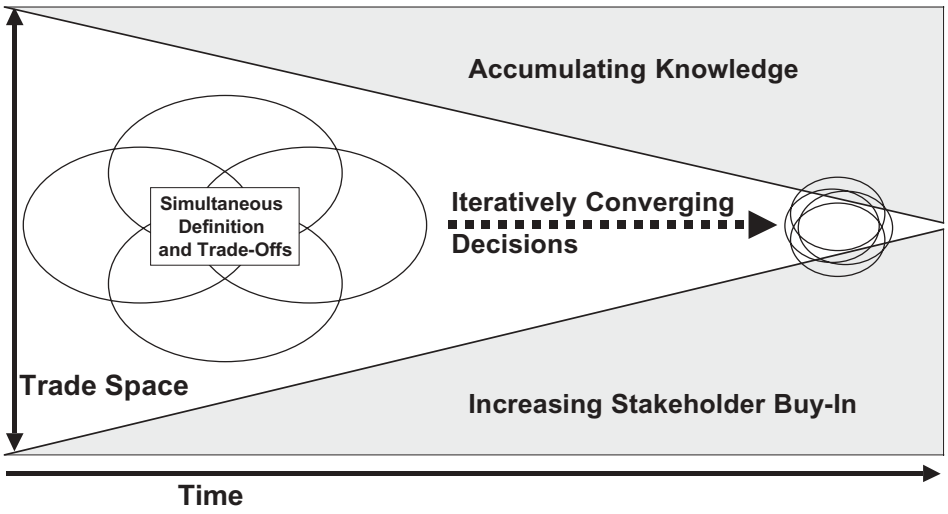
For example, a stakeholder need may be stated such that any known COTS product cannot satisfy it. Similarly, a potential COTS product may not be compatible with the organization's existing infrastructure or use a licensing strategy that would be cost prohibitive.

Therefore, as information among spheres is analyzed, trade-offs among the spheres are identified that must be resolved through negotiation among the disparate stakeholders. In practice, this drives the practitioner to gather a little, synthesize and negotiate a little, and then gather a little more and synthesize and negotiate further. Due to COTS products' volatility, this cycle of gather, synthesize, and negotiate must be repeated until the system is replaced or retired. Further, the new release of an already selected COTS product may change system behavior.

An Evolutionary Integration Process

The EPIC evolved from a U.S. Air Force need to institutionalize a process that implements the necessary simultaneous definition and trade-offs of the required COTS approach. EPIC is documented [2] to provide an overview and detailed instruction. An Air Force organization and a commercial financial institution have started using EPIC across their programs.

Figure 2: *The EPIC Environment*



EPIC does not *simply* evaluate and select COTS products. Rather, it leverages many of the elements of the Rational Unified Process [5] to integrate COTS lessons learned and disciplined spiral engineering practice [6] to define, develop, field, and support COTS-based solutions[4].

The EPIC Framework

To maintain the required balance between the four spheres through the life of the solution, EPIC creates an environment that supports an evolving definition of the solution while systematically reducing the trade space within the spheres. As shown in Figure 2 and discussed in the following paragraphs, this environment consists of iteratively converging decisions and accumulating knowledge while increasing stakeholder buy-in.

Iteratively Converging Decisions Reduce the Trade Space

While trade-offs are common in any engineering endeavor, trade-offs in EPIC are driven by an increasingly detailed knowledge of the COTS products' marketplace capabilities. Initially, as shown at the left of Figure 2, the trade space may be large, with great flexibility for negotiating trade-offs among the four spheres. However, a decision in one sphere influences, and is influenced by, decisions in the other spheres.

Over time, as the stakeholders' understanding of the solution evolves, decisions cause the spheres to converge. As they converge, the spheres become more interdependent, and the available trade space shrinks. Elements of the solution will continue to evolve until the solution is retired as business or operational needs change and new releases of COTS products become available.

Accumulating Knowledge Through Disciplined Risk-Based Spiral Practices

As the trade space diminishes, knowledge about the solution grows more detailed, which is reflected in the set of artifacts or work products necessary to build, field, and support the solution. Most of the artifacts are started in a rough form very early in the process and are expanded as more information is gathered and refined.

Due to marketplace volatility, keeping current knowledge about it is particularly important. In some cases, market events may invalidate already agreed-upon decisions (e.g., support for a product is dropped, a new product is introduced, or a product feature is added). While these disruptions have no easy resolution, relationships with vendors can provide warning of impending changes so appropriate actions can be taken.

Increasing Buy-In Through Continuous Negotiation Among Stakeholders

An environment that includes all affected stakeholders is essential for timely resolution of mismatches between the available COTS products, the desired end-user business processes, and the stated stakeholder needs. In EPIC, *stakeholders* include the broadest set of individuals and organizations affected by the solution (or their empowered representatives). End users, one set of stakeholders, must be involved day-to-day to evaluate each COTS product's impact on the end-user business processes. In addition, the end-user needs will mature and change as their understanding of available COTS products increases. Concurrently, engineering stakeholders ensure that the COTS products considered can be effectively integrated with the organization's existing systems to meet required performance parameters and other system qualities. Business analysts ensure that viable vendors support the products. Vendors, another class of stakeholders, provide enhanced visibility into the COTS products' capabilities and gain potential insight into the organization's needs.

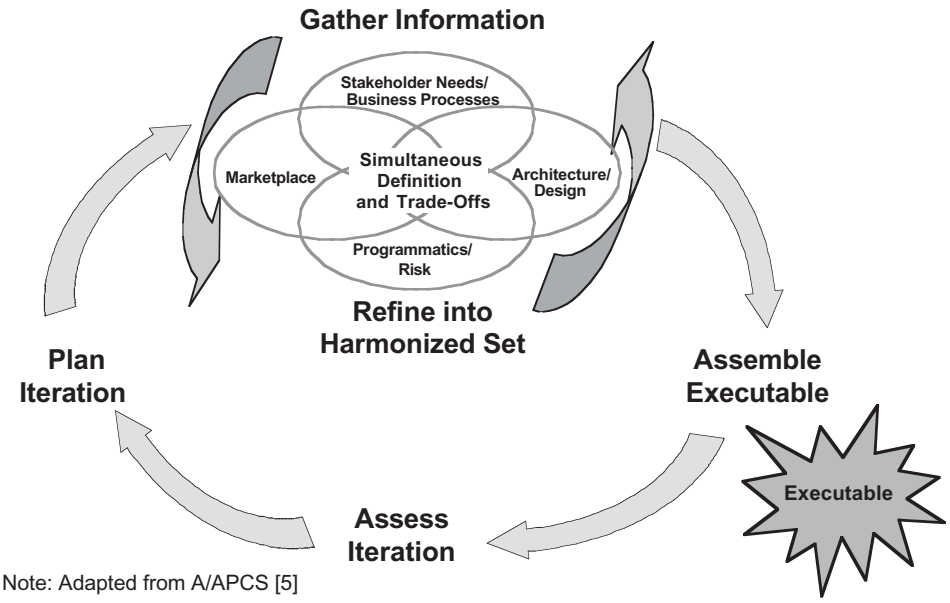
Using EPIC

To implement this environment, EPIC uses a risk-based spiral development process that keeps the requirements and architecture fluid as they are discovered and adjusts them to optimize using COTS products. The following sections summarize the COTS implications on the iteration and phase structure in which project and system activities take place.

Spheres of Influence Are Balanced in Every Iteration

Like other spiral development approaches, a series of iterations is defined across the life of the solution to mitigate specific project risks while addressing the most critical functions. In EPIC, each iteration consists of the fixed set of activities shown in Figure 3.

Each iteration begins with creating a detailed plan to meet defined iteration objectives. While many view the activities within an iteration to be a *mini-waterfall*, with COTS, information must be *simultaneously* gathered from each of the four spheres, as it is refined through analysis and stakeholder negotiation. Due to the interaction among spheres, many cycles of gathering and refining information may be required to produce a consistent set of information across all spheres to meet the iteration objectives. An executable representation of the solution is assembled to demonstrate the current understanding among the stakeholders. The iteration ends with an assessment of whether the objectives were met.



Note: Adapted from A/APCS [5]

Figure 3: *An EPIC Iteration*

Iterations Are Managed Through Clearly Defined Anchor Points

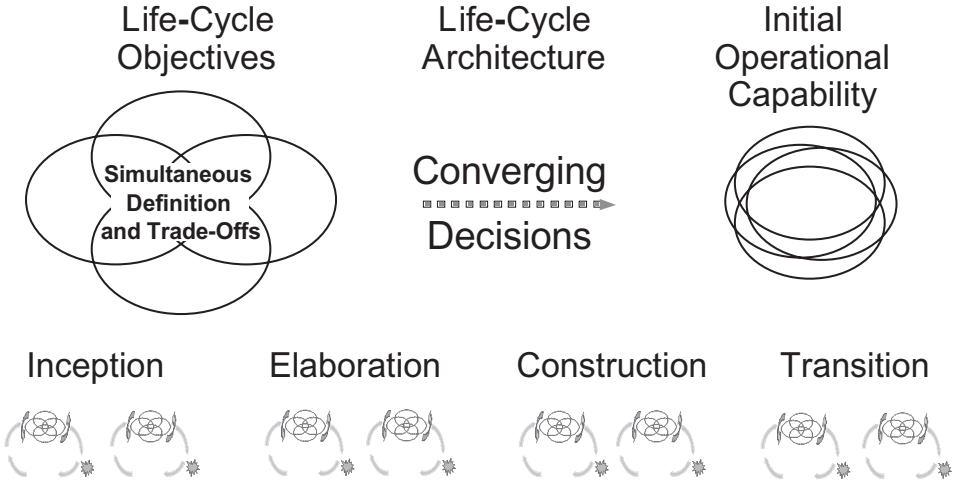
While the activities are the same for each iteration, the focus, depth, and breadth tend to change in character across the life of the system. As shown in Figure 4, EPIC uses the four Rational Unified Process phases (Inception, Elaboration, Construction, and Transition) and three corresponding anchor points (Life-Cycle Objectives, Life-Cycle Architecture, and Initial Operational Capability) to manage spiral development activities across the life cycle.

Like other spiral development processes, each phase consists of one or more iterations designed to meet explicit phase objectives and ends with an anchor point that provides an opportunity to review exit criteria, ensure continued stakeholder commitment, and decide to proceed, change project direction, or terminate the project based on progress to date. The following paragraphs summarize the goal of each phase and highlight some of the changes made in EPIC to accommodate the unique character of COTS-based solutions.

Inception Phase: The Inception Phase establishes a common understanding among stakeholders of what the solution will do and why. It ends with the Life-Cycle Objectives anchor point when it is demonstrated that one or more, albeit high-level, candidate solutions can be integrated into the organization's broad architecture, in a reasonable period of time, at affordable cost, and for acceptable risk. In EPIC, candidate solutions may be formed around substantially different COTS products and, therefore, may address different user processes and stakeholder needs, use different architectures, and have different programmatic implications.

EPIC users have been surprised at the magnitude of the effort necessary to achieve the objectives of this phase. This is due in part to a lack of experience with spiral development processes. Just as significant, however, has been the need to reconcile a variety of

Figure 4: *EPIC Phases*



program mandates, which have included preselection of one or more COTS products by senior leadership, a predefined architectural framework, predefined target business processes, and arbitrary schedule and cost objectives. Identifying and negotiating these issues while the cost of resolving them was low has proven beneficial.

Elaboration Phase: The Life-Cycle Objectives anchor point marks a change in intensity. During the Elaboration Phase, stakeholders conduct in-depth experiments with candidate COTS products in a context that closely represents the operational environment. This phase ends with the Life-Cycle Architecture anchor point when all stakeholders agree that the defined solution provides sufficient operational value; the requirements, end-user business process, and architecture are sufficiently stable; and the solution can be assembled for acceptable cost, schedule, and risk.

While every effort is made to stabilize the solution, inevitably some unanticipated changes will occur in development and maintenance. In particular, new versions of the selected COTS products will require evaluation. Monitoring the marketplace for advance notice of these changed versions is required through the life of the solution.

Some EPIC users have applied this phase differently. One user views elaboration as an opportunity for rapid prototypes. Another uses EPIC to validate the utility of implementing a COTS-based versus custom solution. In both cases, this phase is used to converge on a stable definition of the solution to be implemented and then fielded.

Construction Phase: The Construction Phase focuses on preparing a production quality release of the solution suitable for fielding, and any necessary preparations in the target organizations to facilitate the initial fielding. In addition to developing any custom components, production rigor is applied to tailoring[5] COTS products, developing integrating code (including wrappers, glue, etc.) and fully testing the system. The Construction Phase ends with the Initial Operational Capability anchor point when the solution is of sufficient quality for initial fielding to a subset of operational users.

No organization has yet progressed to this phase using EPIC.

Transition Phase: The Transition Phase fields and supports the solution across the user community. This requires proficiency in using the solution. As required, bugs are fixed, features are adjusted, new COTS product releases and patches are evaluated and considered for integration, and missing elements are added to the fielded solution in maintenance releases. The Transition Phase ends only when the solution is retired and/or replaced by a new solution.

Implications of COTS-Based Systems on CMMI Work Processes

Capability maturity models integrate total quality management, best practices targeted to an organization's domain, and organizational development practices to guide improvement to an organization's processes and its ability to manage the development, acquisition, and maintenance of goods[6] or services. Many organizations have derived substantial benefits [4] from process improvement using capability maturity models.

The CMMI [3] currently contains four disciplines: Systems Engineering, Software Engineering, Integrated Product and Process Development (IPPD), and Supplier Sourcing. Each is critical to building, fielding, and supporting a COTS-based system. Systems Engineering and Software Engineering disciplines provide a basis for the necessary development and maintenance activities. IPPD provides for the timely stakeholder involvement to support negotiations among the spheres of influence. Supplier Sourcing provides practices for selecting and managing subcontractors that may be extended to COTS vendors.

Process areas[7], the primary building blocks for the CMMI, are not a process – nor are they process descriptions. It is intended that practices in the CMMI process areas be interpreted using an in-depth knowledge of the organization, the business environment, and any other relevant circumstances to define and implement work processes that meet the organization's needs.

As illustrated in EPIC, the required COTS-based systems approach demands development and maintenance environments and work processes that support the following behaviors:

- Concurrent definition and evolution of the four spheres:
  - Business process engineering becomes integral to system engineering.
  - Requirements are formed through discovery of what is available in the commercial marketplace and any other sources.
  - A flexible system architecture is defined early and maintained until the system is retired.
  - Awareness of potential marketplace changes is kept current throughout system development and maintenance.
  - Cost, schedule, and risk implications for implementing the system *and* any required business process changes are an integral part of all trades.
- Continuous negotiation among stakeholders.
- Disciplined spiral or iterative practices with frequent executable representations of the evolving system.

In the following sections, selected CMMI process areas are identified for each of these behaviors with guidance for interpreting the process area for COTS-based systems. Not all affected process areas are discussed nor is a full treatment of the needed interpretations provided. Although all process areas are important in a COTS-based system, this article focuses primarily on the project management and engineering process areas, with references to other process areas of particular relevance.

Concurrent Definition and Evolution of the Four Spheres

Development of a COTS-based system is essentially an act of reconciling the four diverse spheres of influence. The discovery of requirements, the solution design, and the formation of project parameters must be fully integrated with stakeholders' discovery and analysis of capabilities in the marketplace. In addition, periodic disruptions to this discovery must be accommodated. This implies the following:

- Decision Analysis and Resolution. Well-established, robust decision processes are required to manage continuous negotiation among the solution's stakeholders.
- Technical Solution. Alternative solutions, including solutions using mixes of different COTS products, must be continuously developed and analyzed to reflect changes across all four spheres.

Business Process Engineering Integral to System Engineering

COTS products implement the vendor's assumptions about how end-user business processes operate. This is very different from a custom development situation where the system is created to meet the demands of predefined operational processes. For COTS-based systems it means that the end-users must be willing to modify their business processes as they learn about candidate COTS products. And, end-user business processes may need to be redetermined and renegotiated with new releases of the COTS product across the life of the system.

The CMMI does not address changes to the processes in the functional units of the enterprise. However, the concepts in Organizational Process Focus can be expanded in application to plan and implement enterprise business or operational

process improvement. In addition, a shared vision of success among stakeholders, suitable incentives, and leadership (as described in the concepts in Organizational Environment for Integration) are critical to aligning business processes with alternative solutions.

Requirements Formed Through Discovery of What Is Available

For COTS-based systems, it is unrealistic to form a detailed set of requirements at the start of a project and force the system to meet that set. Three conditions cause this to be true. First, as previously discussed, to leverage the marketplace, requirements must be informed by an understanding of available COTS products. Second, as end users interact with candidate COTS products and better understand the capabilities in the marketplace their expectations for the solution tend to change. Third, the marketplace changes product capabilities and introduces new technologies that provide unforeseen opportunities and challenges. Thus, requirements need to be fluid enough to respond appropriately to changes in the marketplace across the life of the solution. This implies the following:

- Requirements Development. To aid in making trades, prioritizing the requirements is an essential practice for a COTS-based system. Stakeholders must agree on a minimum set of *must-have* requirements.
- Requirements Management. It is particularly important to begin disciplined and controlled requirements management at project start to track identified and negotiated trades.

Early Definition and Maintenance of a Flexible System Architecture

Since the COTS products are *owned* by the vendors, the framework by which the COTS products and other components of the system are combined to provide desired functionality – the architecture – becomes an important strategic asset. And, evolvability of the system becomes a critical quality attribute.

The architecture must be based on current and predictive knowledge of both the enterprise and the underlying technologies of relevant COTS products, and carefully crafted to insulate parts of the system from changes in other parts. The structure and cohesiveness of the architecture must be maintained while allowing the system to respond efficiently to continuous COTS product upgrades, technology advances, and new operational or business needs until the system is retired. This implies the following:

- Technical Solution. Alternate solutions need to describe the project standards or protocols (often referred to as *glue* code or wrappers) that will be used to link COTS products and other system components. Previous *make-or-buy* decisions may need to be revisited when an existing product changes or a new product becomes available.

Continuous Awareness of Changes in the Marketplace

Relevant market segments and products must be monitored to anticipate and track any changes that could potentially affect the solution. Knowledge of key COTS products must be sufficiently detailed to understand their potential impacts and benefits to the system. Hands-on evaluation of key COTS products and prototypes of combinations of products is essential.

The vendor is an important stakeholder for the project providing unique insights into ways that products work. Developing and maintaining relationships with the vendors who supply key COTS products may help *influence* a vendor's product direction (vendors do not often respond to *direction* from a customer). The specific nature of the relationship will depend on the importance of the COTS product to the solution and the vendor involved. Not all vendors will encourage (or entertain) a close-working relationship. This implies the following:

- Integrated Supplier Management. *Partnering* with key vendors is critical. Vendors, however, will seldom allow process monitoring. Therefore, to determine product suitability, it is important to conduct a hands-on evaluation of each vendor's product releases (including any patches) in the context of their use in the system. In addition, establishing and maintaining relationships to influence (not direct) future product capabilities is critical. Relationships with key vendors' other customers, while not explicitly covered in CMMI, may amplify this influence.

Cost, Schedule, and Risk Implications Integral to All Trades

Each alternative solution must evaluate team skills and expertise required to implement, field, and support it as well as the associated cost, schedule, and risks. In addition, fielding of a COTS-based system includes the cost, schedule, and risks of implementing business process changes for the functional units that are affected by the solution as part of the total cost of ownership. This implies the following:

- Technical Solution. Engineering trades must include the risk, cost, schedule, and other programmatic factors that are associated with each alternative solution. Estimates of work product and task attributes should be generated for each alternative.

Continuous Negotiation Among Stakeholders

Communication and effective decision-making processes are critical to a COTS-based system. Stakeholders who reflect the full diversity of interests must be available to quickly resolve mismatches among elements in the four spheres of influence as they are discovered, and agree that the evolving definition of the system will meet their needs. Stakeholders must be actively involved in the development process, particularly stakeholders who will use the system to meet enterprise objectives.

Integrated teaming among disparate stakeholders (as described in the process areas within the IPPD discipline) throughout development and maintenance is essential. The end users must be involved to confirm the results of any and all negotiations.

Disciplined Spiral or Iterative Practices With Frequent Executables

COTS-based systems are particularly suited to spiral development work processes. In spiral development, critical attributes of the solution are concurrently discovered through an evolutionary and continuous exploration of the highest risk elements of the system. Spiral development encourages frequent and direct feedback from stakeholders while reducing the risk of misunderstandings by producing and validating executable representations (prototypes or production releases) of the evolving solution. This implies the following:

- Project Planning. If not already implemented, extensive effort may be needed to revamp planning and engineering processes to align with a risk-based spiral development approach.
- Risk Management. The risk-management strategy needs to be robust enough to allow risk to (re)direct and manage the project. Tracking the effectiveness of risk mitigation is essential.
- Technical Solution and Product Integration. Frequent executable representations of the evolving alternative solutions provide critical insights into how the solution will operate and how the COTS products will be integrated to achieve essential solution behaviors.

Summary

COTS-based systems introduce unique challenges that demand fully integrated work processes to accommodate the volatility of the marketplace throughout the life of the system. In particular, COTS products and end-user operations must be reconciled – and

re-reconciled – as new product releases become available. This forces linkage between business process, engineering, and system development activities. Extensive communication and strong decision-making processes are necessary to facilitate cooperation, negotiation, and continuous validation of the evolving definition of the solution across potentially disparate stakeholders.

EPIC illustrates a way to realize the promise of COTS using a risk-based spiral development process to actively manage and balance knowledge across four spheres of influence. EPIC is more than a way to select a specific product. Rather, it shows a way to define, develop, field, and support a coherent solution composed of one or more COTS products, any required custom code, and implementation of any changes required to end-user processes.

As with any CMMI application, the unique aspects of COTS-based systems must drive the development of effective work processes. Developing and maintaining a COTS-based system is more than selecting products.

The authors solicit feedback from organizations implementing EPIC or similar processes.◆

References

1. United States Air Force Science Advisory Board. Report on Ensuring Successful Implementation of Commercial Items in Air Force Systems. SAB-TR-99-03. Washington, D.C.: SAB, 2000.
2. Albert, C., and L. Brownsword. Evolutionary Process for Integrating COTS-Based Systems (EPIC): An Overview. CMU/SEI-20030TR-009. Pittsburgh, PA: Software Engineering Institute, 2002 <www.sei.cmu.edu/publications/documents/02.reports/02tr009.html>.
3. Software Engineering Institute. CMMI Product Team: Capability Maturity Model® Integration, Version 1.1. CMU/SEI-2002-TR-11. Pittsburgh, PA: Software Engineering Institute, 2002.
4. Goldenson, D., and J. Herbsleb. After the Appraisal: A Systematic Survey of Process Improvement, Its Benefits, and Factors That Influence Success. CMU/SEI-95-TR-009. Pittsburgh, PA: Software Engineering Institute, 1995.
5. Kruchten, P. The Rational Unified Process: An Introduction. 2nd ed. Addison Wesley Longman, Inc., 2000.
6. Boehm, B. "A Spiral Model of Software Development and Enhancement." IEEE Computer May 1998: 61-72.

Notes

1. A COTS-based system can be one substantial COTS product tailored to provide needed functionality or multiple components from a variety of sources, including custom development, integrated to collectively provide functionality.
2. We use the term *modification* to mean changes to the internals of a hardware device or the software code. This does not include vendor-provided mechanisms for tailoring the product to specific operational environments.
3. A stakeholder is any person or organization with a vested interest in the outcome of a project, including customers, developers, engineers, managers, manufacturers, end-users, etc.
4. In EPIC, a *solution* is the integrated assembly of one or more COTS products or other reuse components, any required custom code (including wrappers and *glue*), appropriate linkage to the organization's broad architecture, and any necessary end-user business process changes.
5. *Tailored* means non-source code adjustment necessary to integrate the COTS products into an operational system, e.g., scripts.
6. Goods are any tangible output intended for delivery to a customer or end user (CMMI uses *product*).
7. To distinguish them from generic process names, CMMI process area names are underlined.

Cecilia Albert and **Lisa Brownsword** will also be speaking at STC 2003 on "Evolutionary Process for Integrating COTS-based systems (EPIC)" on Tuesday, 29 April, Track 8, Room 251 D-F, from 3:00-3:40 p.m.

About the Authors

Barbara Tyson, Ph.D., is a senior member of the technical staff in the Software Engineering and Process Management Group at the Software Engineering Institute. She develops and promulgates software engineering processes and organizational change. Tyson provided software development and systems engineering technical and management support in industry, federal, and academic environments. She has taught graduate courses in marketing, management, organizational change, and information systems at Johns Hopkins and Marymount Universities.

Software Engineering Institute
4301 Wilson Blvd. Suite 902
Arlington, VA 22203
E-mail: btyson@sei.cmu.edu

Cecilia Albert is a senior member of the technical staff in the Commercial Off-the-Shelf-based systems Initiative at the Software Engineering Institute. Albert served in the U.S. Air Force where she developed major software programs for simulation, command and control, and mission processing of national satellite systems. She taught at the Industrial College of the Armed Forces, and managed the archive and dissemination programs at the National Imagery and Mapping Agency.

Software Engineering Institute
4301 Wilson Blvd. Suite 902
Arlington, VA 22203
E-mail: cca@sei.cmu.edu

Lisa Brownsword is a senior member of the technical staff in the Commercial Off-the-Shelf-based systems Initiative at the Software Engineering Institute. Brownsword was on staff at the Computer Sciences Corporation in support of NASA/Goddard's Software Engineering Lab and has worked for Rational Software Corporation where she provided consulting to managers and technical practitioners in the use of transitioning to software engineering practices, including architecture-centered development, product lines, object technology, and Ada.

Software Engineering Institute
4301 Wilson Blvd. Suite 902
Arlington, VA 22203
E-mail: llb@sei.cmu.edu

Planning and Managing the Development of Complex Software Systems

Dr. Richard Bechtold
Abridge Technology

Wednesday, 30 April 2003
Track 4: 1:50 — 2:30
Ballroom D

With the ongoing evolution of information systems and computer technologies, it is becoming progressively easier to leverage incremental design, development, and testing strategies. This article briefly examines the problems inherent in the traditional "grand, all-at-once" implementation approach. Next, an alternative approach is described that leverages grand and incremental design, incremental development, early incremental testing, rapid risk reduction, and the re-calibration of data used for estimation. Key benefits to this approach include easily developed and highly reusable estimation data, early verification of system feasibility, early management of customer expectations, and early validation of system usability and acceptability.

In the initial decades of the computer industry, systems were built in isolation. Large problems typically were solved through the development of numerous, stand-alone, vertical solutions. As computer systems started covering progressively larger segments of the problem space, they began to overlap and often failed to integrate.

The solution to this problem was part systems engineering, and part *grand design* and *grand implementation*. The premise behind these grand approaches was to think of the entire problem and to implement a *total solution*. Although on the surface this seems perfectly reasonable, the problem is that this strategy, especially when coupled with a Waterfall Model life-cycle development approach, simply does not reliably scale up to implementing large, complex systems.

Depending on whose data you reference, there is anywhere from a 50 percent to an 80 percent likelihood that any given software system project will fail [1]. That is, the project will require substantially more time than originally planned, cost substantially more than originally budgeted, or will deliver substantially less functionality than originally expected, or any combination of the preceding. Furthermore, the larger the project, or the longer the planned duration, the greater the likelihood of failure.

After nearly 25 years in the software industry and 10 years of conducting process appraisals using the Software Engineering Institute's Capability Maturity Model® for Software, I have repeatedly observed that it is exceedingly rare for complex, large-scale, multi-year, grand implementation software projects to deliver all expected functionality within the originally planned schedule and budget. But are these failures truly due to the grand implementation approach, or is it something else?

Grand Implementation Problems

In this article, grand implementation approaches are considered to be coupled with a Waterfall Model life-cycle development approach. Although large, complex systems often need a grand design to ensure overall architectural integrity, having such a design does not mean that the next step must be a grand implementation. Nevertheless, after a grand design is completed development often commences for the entire system. When the entire system is ready, it is put though integration test, system test, and acceptance testing: classic Waterfall Model development. While this can be a very effective approach for small systems or short duration projects, it becomes a much less successful approach as project duration and complexity increase.

"Depending on whose data you reference, there is anywhere from a 50 percent to an 80 percent likelihood that any given software system project will fail."

Part of the problem is requirements volatility. The traditional response to this problem is to require the customer to freeze their requirements – as if that were possible. Frozen requirements make absolutely no sense for the simple reason that all systems are basically built to address one root requirement: solve the customer's problems. If a system is being built during several years, what is the likelihood that the customer's problems will remain unchanged throughout this entire period?

I do not know what the aggregate staff-year transition point is regarding when a grand implementation project crosses over from being more likely to succeed to being more likely to fail, but it does not seem like a very large number. Certainly a grand implementation approach to a four-person, three-month project will likely work. Maybe a grand implementation approach can even work on a 10-person, 18-month project if you are really good. But what about a 100-person, five-year, legacy systems modernization project? Can we reliably apply grand implementation thinking to this scale of a project, or should we consider another approach?

There is still another problem with the grand implementation strategy. Even if you can get it to work, you have likely created an absolute nightmare for whoever will build the eventual replacement system. Too often, people who design systems do not think about an incremental approach to that system's *retirement*. If you do not design the system for incremental retirement, then in all likelihood you will not be able to conduct an incremental retirement.

Typically, this forces the replacement system to also be based upon (yet another) grand implementation, *all-or-nothing* solution. Replacement systems are usually much more complex than the systems they replace, so the problem of *total solu-*

tion replacement – often referred to as *legacy system modernization* – can lead to projects that are progressively more likely to repeatedly fail [2].

Designing and Planning Complex Software Systems

The remaining sections of this article look at a systematic alternative approach for complex systems' construction. For simplicity, this discussion generally focuses on the design, estimation, planning, and management of the construction and delivery of complex Web-based systems or systems that include Web-based subsystems, but the principles are also applicable to the construction of other types of software-intensive systems.

Micro-Incremental Development

Incremental development is not a new concept, but the eXtreme Programming (XP) community has given further definition to some of the principles. In particular, XP advocates using a highly incremental approach, and building testable functionality in much smaller and shorter duration steps [3]. The approach advocated in this article adds additional details to this foundation.

When commencing system decomposition, strive to allocate high-level functionality into regions, partitions, and frames. For this article, regions are defined as major subsystems that can be separately implemented, tested, maintained, and replaced. At the design level, regions capture required system functionality but defer physical implementation details.

Each region consists of several partitions that can be separately implemented, tested, maintained, and replaced. At the design level, partitions reflect not only functional requirements but also capture all important physical implementation details.

Lastly, each partition is divided into several frames. Frames are low-level or atomic software components such as Java class files that can be separately assigned to small software teams for parallel construction and unit testing.

Generally, a large system can be designed with five to 10 distinct regions, each with five to 10 distinct partitions. Completed regions should deliver actual usability to the customer or system end-users. Within each region, completed partitions should deliver actual functionality. Although functionality does not necessarily translate into usability, it does allow the customer or system end-users to evaluate system characteristics and performance, and to provide early feedback to the development team [3].

Hot-Swappable Partitioning

One of the key tenants of this approach is that each partition is *hot-swappable* or capable of being replaced with little or no adverse impact on the rest of the system. Given the preceding guidelines of five to 10 regions, each with five to 10 partitions, your design will contain between 25 and 100 separately implementable, testable, and replaceable partitions. To facilitate this, partitions can be designed to interact with each other primarily through message passing and file input/output.

"Even if you can get it [grand implementation strategy] to work, you have likely created an absolute nightmare for whoever will build the eventual replacement system ... you will not be able to conduct an incremental retirement."

Of course, there are trade-offs to be addressed. Partitions that communicate via file input/output will suffer a severe performance penalty. Additionally, as systems evolve and grow larger, there is a general tendency for them to become increasingly interconnected. Avoiding this will usually require periodic efforts to reduce partition coupling and to increase partition cohesion, both of which translate into increased time, money, and effort. However, by taking steps to ensure hot-swappable partitioning there is an increased likelihood that you can more easily fix or upgrade individual partitions without adversely impacting the rest of the system.

Nevertheless, given the preceding issues it is clear that 100 percent hot-swappable partitioning is more of an ideal than it is a practical reality. At a minimum, however, it is certainly critical to avoid the *all-or-nothing* implementation where the system is either completely working or completely useless. Within the limits of system performance, project budgets, and schedule constraints, hot-swappable partitioning should be a top-priority design objective and built into as many partitions as possible.

Commitment Deferral

A third key concept is the deferral of technical commitments, and especially architectural commitments, to the greatest extent possible. Again, in some areas commitment deferral may be impossible or, indeed, not even desirable. Nevertheless, during design you may have some opportunities to allocate certain technologies to partitions that you plan to build towards the end of the project. By deferring commitments you increase the likelihood that you can more easily respond to changing requirements, changing technologies, or evolving solution alternatives.

Early Detection/Agile Response

Further leveraging XP, a fourth key concept is to plan for a development and implementation approach that allows for early detection of any problems, and for agile and rapid response to those problems. Given that you have designed for both usable releases (regions) and functional releases (partitions), system components can be developed, tested, and immediately delivered to key stakeholders. In addition to early detection and correction of defects and improved customer communications, this approach will allow you to perform highly effective customer expectation management.

Partition and Frame Estimation

Even though the preceding guidelines will result in a system where functionality is implemented across 25 to 100 partitions, these partitions may still be too large to accurately estimate required work. Therefore, as indicated earlier, each partition can be further decomposed into five to 10 frames. The primary objective of frames is to facilitate planning, construction, unit testing and integration testing, and to support risk management (more on this later).

Each frame should be assigned to one person or to a very small team (such as with pair programming). Additionally, each frame should generally appear to require somewhere between one and four weeks of work. When it is obvious that less work will be required, frames can be combined. Conversely, if more work is obviously required, frames can be further

decomposed.

The next step is to analyze the key attributes of each frame and to use those attributes as the basis for schedule and cost estimation. Although key attributes will vary substantially between different software systems, some attributes will be almost universally important, such as a system's diagnostic capability [4].

For example, the following are five key attributes to consider for systems that include Web-based partitions:

- *Artwork* or static content.
- *Logic* or core dynamic content.
- *Diagnostics* or the ability to detect misuse or system intrusions.
- *Security* or the ability to prevent misuse or system intrusions.
- *Containment and Recovery* or the ability to perform damage control and repair.

As shown in Figure 1, each attribute is analyzed and rated using a five-block by five-block grid. To rate the preceding five different attributes, you would use five separate tables. Figure 1 shows an empty grid for estimating required security for a particular frame.

The rows of the grid indicate the relative amount of work to be done. Rows are labeled top-down from *E* to *A* indicating an extremely high amount of work to an extremely low amount of work, respectively. The columns of the grid indicate the relative complexity of the work to be done. Columns are labeled left to right from 1 to 5 indicating extremely low complexity to extremely high complexity, respectively. Since the outermost rows and columns represent extremes, most attributes should be rated in rows B, C, or D and in columns 2, 3, or 4.

As each of the five grids is completed, the estimator determines a confidence level of low, medium, or high and then documents this in the upper left corner. In Figure 2, the estimator indicated they had a low level of confidence in their estimate and that security complexity is rather low (column 2) for this frame. However, the amount of work related to implementing security features is shown as extremely high (row E).

After the five key attribute grids are done, the estimator uses them as *support information* for the composite grid. The composite grid, as shown in Figure 3, is used to estimate overall staff-days for constructing a frame. The cells of the composite grid contain values representing the expected staff days, and the estimator simply circles one of the values. Note that there is a general, *but not algorithmic*, relation between the key attribute basis grids and the composite grid. For example, if most of the basis grids were marked in the upper right regions, it would normally occur that the composite grid would likewise have a cell selected from the upper right region.

As a final step, the estimator uses the upper left square in the composite grid to indicate low, medium, or high confidence in the accuracy of their selection for expected staff-days duration.

The default values shown in Figure 3 were deliberately selected so that *normal* work would span from one to three weeks (the interior three rows and columns). Recall that the work for each frame was initially intended to be *obviously* between one and four weeks. The default values in the composite grid actually accommodate ranges from 40 percent of the obvious minimum to 150 percent of the obvious maximum, or work that spans from two days (A1) to six weeks (E5).

Given these default values, project durations can span from approximately one staff-year (five regions, each with five partitions, each with five frames, each estimated at two days) to approximately 120 staff-years (10 regions, each with 10 partitions, each with 10 frames, each estimated at six weeks). Decomposition can be reduced, or increased, to accommodate shorter or longer project durations, respectively. Similarly, the default values in the composite table can – and indeed should – be adjusted over time to better reflect your actual projects and performance [5].

When you have completed these design and planning steps, you will have a comprehensive and detailed foundation from which to commence managing and controlling your project.

Security					
	1	2	3	4	5
E					
D					
C					
B					
A					

Figure 1: *Key Attribute Analysis*

Security					
Low	1	2	3	4	5
E		X			
D					
C					
B					
A					

Figure 2: *Example Ranking*

Composite (Staff Days)					
	1	2	3	4	5
E	5	10	15	20	30
D	4	10	15	15	20
C	3	5	10	15	15
B	3	5	5	10	10
A	2	3	3	4	5

Figure 3: *Composite Grid*

Managing and Controlling Complex Software Systems Development

As mentioned previously, one of the objectives of the third level decomposition (the frame level) is to support risk management. To accomplish this, it is recommended that you commence actual development with the easiest two regions. Within each of those regions, commence development on the easiest two partitions. Within each of those partitions, commence work on the easiest two frames. The objective of this approach is twofold. First it helps ensure that the development team's learning curve occurs in the least challenging parts of the overall system. Second, you quickly accomplish finished frames.

When you are done with the easiest frames, then commence work on what appear to be the hardest frames. Likewise, commence work in the same pattern with partitions and regions. The objective here is – after much of your learning curve is behind you – to reduce project risk as rapidly as possible. By rapidly undertaking the hardest or most challenging parts of the system, you can more quickly discover whether or not there are any insurmountable hurdles. On the outside chance that you may have to revisit your design, or may need to resort to an alternative solution, these recovery steps are happening much earlier than they might otherwise. Because they are occurring earlier, and because, as discussed earlier, one of your design goals was also to defer commitments even in the event that you need to take an alternate approach, you have a much greater likelihood that you will be able to keep a

greater percentage of the system already developed.

Another benefit to first doing the easiest work, then doing the hardest work, and then doing the average work is that data collected during development rapidly becomes directly usable to recalibrate your estimations and to more accurately predict the remaining work on the project. Depending on the total number of frames, when you are as little as 20 percent into the project you may have a highly reusable set of actual data relating to frames built, estimated time to build them, and actual time to build, test, and debug. This data can be used to further improve the accuracy of estimates relating to the rest of the project.

An important part of this approach is to take a few moments after a frame is completed and do a retrospective estimation worksheet. The format for these worksheets is identical to the original worksheets – the only difference is the time they are completed. By taking this approach, each frame will ultimately have a minimum of two estimation worksheets. One worksheet was done before work commenced, and another one was done after work was completed. By comparing these before and after estimation worksheets, you can analyze and adjust your estimation approach if, for example, you see a clear tendency to underestimate either the amount or complexity of work relating to one or more of the key attribute types.

As you develop the system, to the greatest extent possible strive to rapidly deliver partitions to the primary stakeholders. This will allow for stakeholder evaluation and feedback, and for you to take a proactive approach when addressing conflicting expectations among the stakeholders. Ideally, you can provide early access and insight to end users, program managers, procurement specialists, subject-matter experts, and anyone else who may have a strong and influential opinion about the usability and acceptability of the final system.

Lastly, when tracking and reporting progress, perform only binary accounting at the frame level. That is, do not ask developers for a percent complete estimate on a given frame (we all know what they will tell you). Instead only ask, "Have you started working on it?" and "Have you finished working on it?" Before a frame is completely done, its *percent complete* is zero. After a frame is completely done, its percent complete is 100. At the partition and region level you can easily calculate and report actual percent complete as a function of the percentage of completed underlying frames.

Conclusions

Grand implementation solutions and the Waterfall Model life cycle were perfectly acceptable approaches during the 1970s and even the 1980s. However, as systems continue to become progressively more distributed and exponentially more complicated there are significant opportunities to deliberately design these systems so that they can be incrementally constructed, incrementally tested, incrementally delivered, and incrementally evaluated. This approach directly supports early verification of system feasibility and early validation of system usability and acceptability.

Arguably, at least as important as the preceding benefits, this approach also directly supports the creation of systems that can be incrementally upgraded, incrementally retired, and ultimately incrementally replaced.

Acknowledgments

Various individual aspects of this approach have been in use for years on systems and software engineering projects, XP projects, Capability Maturity Model®-compliant projects, and elsewhere. However, it is hoped that this article presents a new and integrated view of these various individual best practices, and combines them in a way that will provide substantial and reliable risk reductions.

The design, planning, development, and management life cycle described in this article, i.e., *micro-incremental* implementation and evaluation, *hot-swappable partitioning*, *commitment deferral*, *early detection/agile response*, and *partition and frame estimation* is something I have advocated for years. However, the technique for basis estimation of key attributes, and then using that as inputs to composite tables, occurred to me while I was listening to a presentation being given by Rita Hadden on "Credible Estimation for Small Projects" at the 1st International Conference on Software Process Improvement in Washington, D.C., in November 2002. I'm not exactly sure what the specific connection is, but someone else might see one. In any event, her presentation was not only practical and informative, but also a source of inspiration.◆

References

1. The Standish Group. Chaos. Boston, MA: The Standish Group, 1994.
2. Bechtold, Richard. The Fatal Flaw of the Information Systems Industry: Failing to Design for Incremental System Retirement. Accepted for Proc. of the Project Management Institute Seminar, PDS 2003. San Antonio, TX, June 2003.
3. Beck, Kent, and Martin Fowler. Planning eXtreme Programming. Addison-Wesley, 2001.
4. Bechtold, Richard. Diagnostic Software Architectures. Proc. of the Second International Workshop on Development and Evolution of Software Architectures for Product Families. Las Palmas de Gran Canaria, Spain, Feb. 1998.
5. Bechtold, Richard, and Patricia Larsen. Planning and Estimating Complex Web-Based Projects. Proc. of the Software Engineering Process Group Conference. Boston, MA, Feb. 2003.

About the Author

Richard Bechtold, Ph.D., is president of Abridge Technology. He is an independent consultant who assists industry and government with organizational change and systematic process improvement, especially in the area of implementing effective project management. Bechtold has nearly 25 years of experience in the design, development, management, and improvement of complex software systems, architectures, processes, and environments. This experience includes all aspects of organizational change management, process appraisals, process definition and modeling, workflow design and implementation, and managerial and technical training. Bechtold also teaches graduate-level courses in software project management, systems analysis and design, principles of computer architectures, and object-oriented Java programming at George Mason University. The second edition of his latest book, "Essentials of Software Project Management," is scheduled for publication in 2003.

Abridge Technology
42786 Oatyer Court
Ashburn, VA 20148-5000
E-mail: rbechtold@rbechtold.com

Interface-Driven, Model-Based Test Automation

Dr. Mark R. Blackburn, Robert D. Busser, and Aaron M. Nauman
Software Productivity Consortium

Thursday, 1 May 2003
Track 8: 9:15 - 9:55
Room 251 D - F

This article describes an interface-driven approach that combines requirements modeling to support automated test-case and test-driver generation. It focuses on how test engineers can develop more reusable models by clarifying textual requirements as models in terms of component or system interfaces. The focus of interface-driven modeling has been recognized as an improvement over the process of requirements-driven, model-based test automation. Model-based testing users identified the insights and recommendations for applying this approach when they began to scale their model-based testing efforts on larger projects.

Model-based test automation has helped reduce cost, provide early identification of requirements defects, and improve test coverage [1, 2, 3, 4, 5, 6]. Industry use of model-based test automation has provided insight into practical methods that use interface-driven analysis with requirements modeling to support automated test generation. However, the term interface is used loosely in this article.

An *interface* is a component's inputs and outputs, along with the mechanism to set inputs, including state and history information, and retrieve the resulting outputs. Recommendations are provided to perform the modeling of textual requirements in conjunction with interface analysis to support reuse of models and their associated test driver mappings.

Test-driver mappings specify the relationships between model variables and the interfaces of the system under test. The insights are useful for understanding how to scale models and the associated test-driver mappings to support industry-sized verification projects, while supporting organizational integration that helps leverage key resources.

We have applied the model-based test-automation method called the Test Automation Framework (TAF) since 1996. TAF integrates various government and commercially available model-development and test-generation tools to support defect prevention and automated testing of systems and software. TAF supports modeling methods that focus on representing requirements such as the Software Cost Reduction (SCR) method as well as methods that focus on representing design information such as Unified Modeling Language (UML)-based tools or Mathwork's Simulink, which supports control system modeling for automotive and aircraft systems.

With model translation, requirements-based or design-based models are converted into test specifications. T-VEC is the test generation component of TAF that uses the test specification to produce tests. T-VEC supports test-vector generation, test-driver generation, requirements test-coverage analysis, and test results checking and reporting. *Test vectors* include inputs as well as the expected outputs with requirement-to-test traceability information. The test-driver mappings and the test vectors are inputs to the test-driver generator that produces test drivers. The test drivers are then executed against the implemented system during test execution.

There are papers that describe requirements modeling [7] and others with examples that support automated test generation [3, 8, 9, 10]. Aissi provides a historical perspective on test-vector generation and describes some of the leading commercial tools [11]. Pretschner and Lotzbeyer briefly discuss extreme modeling that includes model-based test generation [12], which is similar to uses of TAF. There are various approaches to model-based testing and Robinson hosts a Web site that provides useful links to authors, tools, and papers [13].

Why Interface-Driven Modeling?

It may seem appropriate first to develop models from the requirements, but when developing models for testing purposes, the models should be developed in conjunction with analysis of the interfaces to the component or system under test. Modeling the behavioral requirements is usually straightforward and easier to evolve once the interfaces and operations are understood because the behavioral requirements, usually defined in text, must be modeled in terms of variables that represent objects accessible through interfaces.

A *verification model* is a refinement of the requirements specified in terms of the component's interfaces. Verification modeling from the interfaces is analogous to a test engineer developing tests in terms of specific interfaces of the component under test. The test engineer's role involves developing verification models from requirements. The requirements engineers, sometimes synonymous with system engineers, continue to develop textual requirements as well as any other type of analytical model. Design engineers should focus on identifying the components of the system architecture, and provide the component interfaces to the test engineer.

These requirements and interfaces can then be used by test engineers to construct formalized verification models (in a tool like SCR). The test engineer can use TAF tools to perform model analysis and correct any inconsistency, as well as produce test vectors and test drivers. Once completed, the textual requirements and models, as well as the formalized verification models are passed to the designers and implementers.

Once code is created, the generated test drivers can be used to test the implementation. TAF translators convert verification models into a form where the T-VEC system generates test vectors and test drivers, with requirement-to-test traceability information that allows failures to be traced backwards to the requirement. The designer/implementer can then use the test drivers to test the implemented system, continuously and iteratively.

Figure 1 on page 28 provides a perspective of the verification modeling process flow. A modeler is supplied with various inputs. Although it is common to start the process with poorly defined requirements, inputs to the process can include requirement system and software specifications (e.g., System Requirements Specification [SRS], Software Requirements Specifications [SWRS]), user documentation, interface control documents, application program interface documents, previous designs, and old test scripts.

A verification model is composed of a model and one or more test-driver mappings. A test driver consists of object mappings and a schema (pattern). Object mappings relate the model objects to the interfaces of the system under test. The schema defines the algorithmic pattern to carry out the execution of the test cases. Models are typically

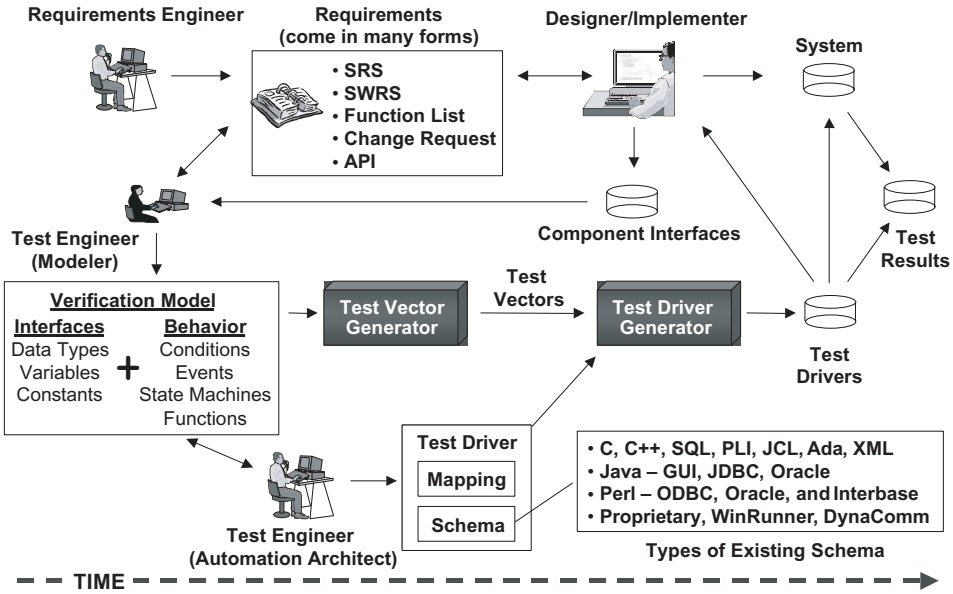


Figure 1: *Interface-Driven, Model-Based Test Automation Supports Continuous Verification and Validation*

developed incrementally. The test-vector generator also detects non-testable modeled requirements (i.e., requirements with contradictions).

Table-based modeling, like the SCR method, has been effective and relatively easy to learn for test engineers [2]. Design engineers commonly develop models based on state machines or other notations such as the UML. However, users and project leaders observed that test engineers found it easier to develop requirements for test in the form of tables (See [8] for details). The modeling notations supported by tools for the SCR method have well-defined syntax and semantics allowing for a precise and analyzable definition of the required behavior.

What Are Some Modeling Perspectives?

Specification languages, usually supported through graphical modeling environments, describe models. Specification languages provide abstract descriptions of system and software requirements and design information. Cooke et al. developed a scheme that classified specification language characteristics [14]. Independent of any specification language, Figure 2 illustrates three specification categories based on the purpose of the specification. Cooke et al. indicate that most specification languages usually are based on a hybrid approach that integrates different classes of specifications.

Requirements specifications define the boundaries between the environment and the system and, as a result, impose constraints on the system. Functional specifications define behavior in terms of the interfaces between components, and design specifications define the component itself. A verification model, in the context of this article, is best classified as a functional specification.

Design-for-Test Supports Interface Accessibility

It is best to understand the interfaces of the system under test before modeling the behavioral requirements to ensure that the interfaces for the resulting test-driver map to actual inputs or outputs of the system under test. If the interfaces are not formalized or completely understood, requirements models can be developed, but associated object mappings required to support test-driver generation must be completed after the interfaces have been formalized.

This can make the object-mapping process more complex because the model entities may not map to the component interfaces. In addition, if the component interfaces are coupled to other components, the components are typically not completely controllable through separate interfaces. This too can complicate the modeling and testing process. Consider the following conceptual representation of the set of components and interfaces shown in Figure 3.

There is a specific way to support a systematic verification approach that can be performed in stages where each component is completely verified with respect to the requirements allocated to it. The interfaces to the component should be explicitly and completely accessible, either using global memory or, better, through get-and-set methods/procedures in Figure 3.

For example, if the inputs to the B.2 component of higher-level component B are completely available for setting the inputs to B.2, and the outputs from the B.2 functions can be completely observed, then the functionality within B.2 can be completely specified and systematically verified. However, if interfaces from other components such as B.1 are not accessible then some of the functionality of the B.2 component is coupled with B.1, and the interfaces to B.2 must also include interfaces to B.1 or to other upstream components such as component A. This interface coupling makes the test-driver interfaces more complex to describe, but also forces the behavioral modeling to be described in terms of functionality allocated to combinations of components.

The coupling reduces components' reuse and increases the regression testing effort due to the coupled aspects of the system components. The problems associated with testing highly coupled systems can be problematic for model-based testing and also negatively affect any type of testing. We have observed that interface-driven modeling has helped foster better system design by reducing the coupling, but also has helped provide better testing support.

Understanding Interfaces

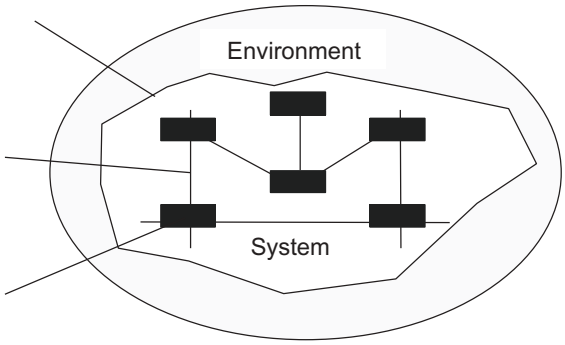
One of the most noted results of the TAF was its application to the Mars Polar Lander (MPL). NASA launched the MPL project on Feb. 7, 1994. Six years later on Dec. 3, 1999, after the MPL had traveled more than 35

Figure 2: *Specification Purposes*

Requirements Specification: Defines the boundary between the environment and the system.

Functional Specification: Defines the interfaces within the system.

Design Specification: Defines the component.



Note: D. Cooke et al., 1996

million miles and was minutes away from its scheduled landing, all contact with the craft was lost. The MPL cost $165 million to develop and deploy.

After the fact, we had the opportunity to use the TAF to see if it would have found the bug. We deliberately did not look at the code before creating our tests. Instead, we created them by modeling the English-language requirements in a tool based on the SCR method. We modeled the Touchdown Monitor (TDM) requirements using the TAF tools and were able to identify the software error associated with the MPL's landing procedures in fewer than 24 hours.

The TDM is a software component of the MPL system that monitors the state of three landing legs during two stages of the descent. As shown in Figure 4, a real-time multi-tasking executive calls the TDM module at a rate of 100 times per second and receives information on the leg sensors from a second module. These two modules establish the interfaces to TDM. During the first stage, starting approximately five kilometers above the Mars surface, the TDM software monitors the three touchdown legs.

There is one sensor on each leg to determine if the leg touched down. When the legs lock into the deployed position, there was a known possibility that the sensor might indicate an erroneous touchdown signal. The TDM software was to handle this potential event by marking a leg that generates a spurious signal on two consecutive sensor-reads as having a bad sensor. During the second stage, starting about 40 meters above the surface, the TDM software was to monitor the remaining *good* sensors. When a sensor had two consecutive reads indicating touchdown, the TDM software was to command the descent engine to shutdown.

There is no absolute way to confirm what happened to the MPL. It is believed that one or more of the sensors did have two consecutive reads before the 40 meter point; the leg-sensor information was stored in TDM program memory. When the MPL crossed the 40-meter point, the TDM changed states and read the memory associated with the leg-sensors during the first stage of descent. Because the memory indicated two consecutive reads engineers believe that the engine thruster was shut off at about 40 meters above the Mars surface. Developers could have designed and implemented the requirement in many ways, but the essence of the design flaw is that the program variables retained the state of the bad sensor information.

Organizational Best Practices

Interface-driven modeling can be applied after development is complete; however, significant benefits have been realized when it is applied during development. Ideally, test engineers work in parallel with developers to stabilize interfaces, refine requirements, and build models to support iterative test and development. Test engineers write the requirements for the products (which in some cases are poorly documented) in the form of models, as opposed to hundreds or thousands of lines of test scripts. They generate the test vectors and test drivers automatically.

During iterative development, if the component behavior, the interface, or the requirements change, the models are modified and test cases and test drivers are regenerated and re-executed. The key advantages are that testing proceeds in parallel with development. Users like Lockheed Martin state that test is being reduced by about 50 percent or more, while describing how early requirement analysis significantly reduces rework through elimination of requirement defects (i.e., contradiction, inconsistencies, and feature interaction problems) [2, 15].

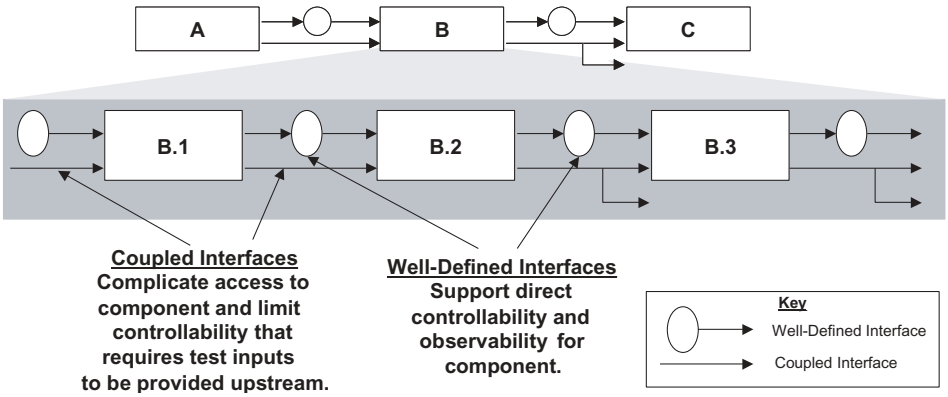


Figure 3: *Conceptual Components and Interfaces of a System*

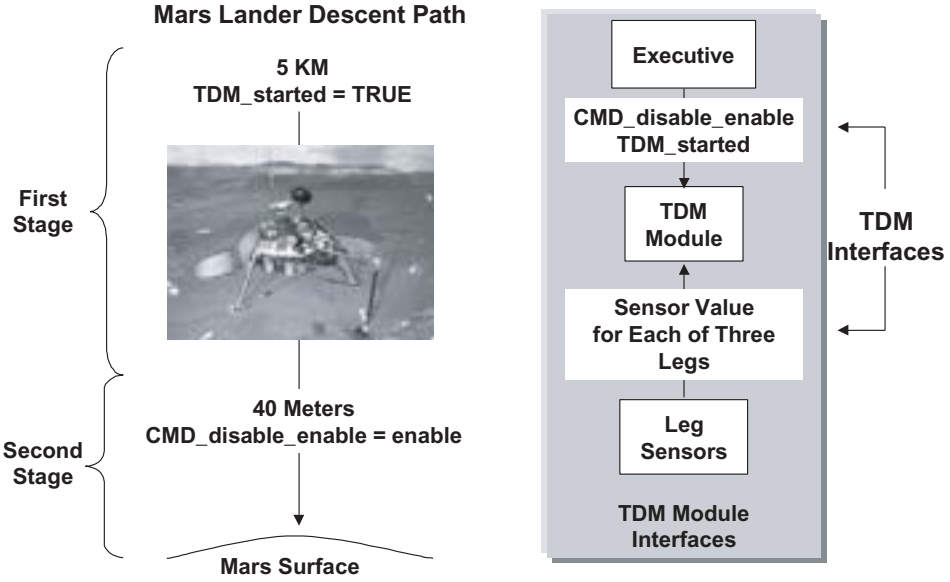
Other Applications

TAF has been applied to applications in various domains, including critical applications for aerospace, medical devices, flight navigation, guidance, autopilots, display systems, flight management and control laws, engine controls, and airborne traffic and collision avoidance. TAF has also been applied to noncritical applications like databases, client-server, Web-based, automotive, and telecommunication applications. The related test-driver generation has been developed for many languages (e.g., C, C++, Java, Ada, Perl, PL/I, SQL, etc.) as well as proprietary languages, COTS test-injection products (e.g., DynaComm, WinRunner) and test environments. Most users of the approach have reduced their verification/test effort by 50 percent [2, 15].

Summary

This article provides pragmatic guidance for combining interface analysis and requirements modeling to support model-based test automation. The model-based testing method and tools described here have been demonstrated to significantly reduce cost and effort for performing testing, while also being demonstrated to identify requirements defects that reduce costly rework.

Figure 4: *Mars Polar Lander Details*



These recommendations for defining interfaces that provide better support for testability are valid for all forms of testing. Organizations can see the benefits of using interface-driven, model-based testing to help stabilize the interfaces of the system early, while identifying common test-driver support capabilities that can be constructed once and reused across related tests. Finally, parallel development of verification modeling is beneficial in development and helps identify requirement defects early to reduce rework.

The TAF is a framework that integrates various commercial and government tools, however, the Software Productivity Consortium does not provide licenses for these tools. For additional information on obtaining these tools, contact the authors. Finally, many referenced papers in the article can be downloaded from the Software Productivity Consortium Web site <www.software.org/pub/taf/Reports.html>.◆

References

1. Rosario, S., and H. Robinson. "Applying Models in Your Testing Process, Information and Software Technology." 42.12 (1 Sept. 2000).
2. Kelly, V., E. L. Safford, M. Siok, and M. Blackburn. "Requirements Testability and Test Automation," Lockheed Martin Joint Symposium, June 2001.
3. Blackburn, M. R., R. D. Busser, A. M. Nauman, R. Knickerbocker, and R. Kasuda. "Mars Polar Lander Fault Identification Using Model-Based Testing." Proc. in IEEE/NASA 26th Software Engineering Workshop, Nov. 2001.
4. Busser, R. D., M. R. Blackburn, and A. M. Nauman. "Automated Model Analysis and Test Generation for Flight Guidance Mode Logic." Digital Avionics System Conference. Indianapolis, IN, 2001.
5. Statezni, David. "Test Automation Framework, State-Based, and Signal Flow Examples." Twelfth Annual Software Technology Conference. Salt Lake City, UT, 30 Apr.-5 May 2000.
6. Statezni, David. "T-VEC's Test Vector Generation System." Software Testing and Quality Engineering. May/June 2001.
7. Heitmeyer, C., R. Jeffords, and B. Labaw. "Automated Consistency Checking of Requirements Specifications." ACM TOSEM 5.3 (1996): 231-261.
8. Blackburn, M. R., R. D. Busser, and A. M. Nauman. "Removing Requirement Defects and Automating Test." STAR-EAST. Orlando, FL, May 2001.
9. Blackburn, M. R., R. D. Busser, and A. M. Nauman. "Eliminating Requirement Defects and Automating Test." Test Computer Software Conference, June 2001.
10. Blackburn, M. R., R. D. Busser, A. M. Nauman, and R. Chandramouli. Model-Based Approach to Security Test Automation. Proc. of Quality Week, June 2001.
11. Aissi, S. "Test Vector Generation: Current Status and Future Trends." Software Quality Professional 4.2 (Mar. 2002).
12. Pretschner, A., and H. Lotzbeyer. Model-Based Testing with Constraint Logic Programming: First Results and Challenges. Proc. of 2nd ICSE Intl. Workshop on Automated Program Analysis, Testing, and Verification. Toronto, Canada, May 2001.
13. Robinson, H. Model-Based Testing <www.model-based-testing.org>.
14. Cooke, D., A. Gates, E. Demirors, O. Demirors, M. Tankik, and B. Kramer. "Languages for the Specification of Software." Journal of Systems Software 32 (1996): 269-308.21.
15. Safford, Ed L. "Test Automation Framework, State-Based and Signal Flow Examples." Twelfth Annual Software Technology Conference. Salt Lake City, UT, 30 Apr.-5 May 2000.

About the Authors

Mark R. Blackburn, Ph.D., is a Software Productivity Consortium fellow with 20 years of software systems engineering experience in development, management and applied research of process, methods, and tools. He has made more than 30 presentations at conferences and symposia, and is involved in consulting, strategic planning, proposal and business development, as well as developing and applying methods for model-based approaches for requirement defect removal and test automation.

Software Productivity Consortium
2214 Rock Hill Road
Herndon,VA 20170
Phone: (703) 742-7136
Fax: (703) 742-7350
E-mail: blackbur@software.org

Robert D. Busser is a principal member of the technical staff of the Software Productivity Consortium. He has more than 20 years of software systems engineering experience in development, and management in the area of advanced software engineering, and expertise in software engineering processes, methods and tools. He has extensive experience in requirement and design methods, real-time systems, model-based development and test generation tools, model analysis, and verification.

Software Productivity Consortium
2214 Rock Hill Road
Herndon,VA 20170
Phone: (954) 753-9634
Fax: (703) 742-7350
E-mail: busser@software.org

Aaron M. Nauman is a senior member of the technical staff of the Software Productivity Consortium. He has a wide range of systems and applications development experience in both real-time and information systems domains. Nauman is involved in the development of model-based transformation tools for automated model analysis and test generation. He has experience in object-oriented technologies, distributed and client/server systems, Web-based and components-based software and systems integration.

Software Productivity Consortium
2214 Rock Hill Road
Herndon,VA 20170
Phone: (703) 742-7104
Fax: (703) 742-7350
E-mail: nauman@software.org

Deployment: Moving Technology Into the Operational Air Force

Lt. Col. Scott B. Dufaud (Ret.)
Software Engineering Services, Inc.

Dr. Lynn Robert Carter
Software Engineering Institute

Monday, 28 April 2003
Track 3: 4:00 - 4:40
Ballroom C

Just purchasing and installing new technology does not bring lasting mission value; the real tasks begin only after the acquisition is completed. If we expect to get the full measure of value from our technologies, we must pay close attention to a whole host of issues to ensure that daily operations are enhanced to obtain full utilization of our technology assets.

There are some things America's armed services do really well. No other military force in the world projects resources and power to the battlefield the way we do. We train for wars and disasters realistically, and then we execute the way we trained. We demonstrate the realm of possibilities in our spectacular air shows and say to everyone, "This awe-inspiring display of teamwork shows what training and discipline can accomplish. Would you like to be part of something much larger than just you?" We are able to deliver when it really counts despite a host of problems and challenges, be they problems with new technology, problems communicating, or the most common of all, problems that come from being human.

There are, however, some areas where improvement would be helpful and within the realm of possibility. More practice understanding, communicating, and coordinating could reduce parochial behavior between stovepipe and functional communities. We have wonderfully effective teams, yet many leaders and team members fail to appreciate what it takes to form a truly high-performance team that can sustain consistent execution at what often appears to be superhuman levels.

Oftentimes, we hold too closely to the paths and the techniques that got us where we are and fail to embrace change when it is truly a win-win proposition. In the domain of software-intensive technological systems, which form the core of our military capabilities, our warriors and those who support them often face the prospect of changes to the status quo. Unfortunately, we are not always as successful as we could be. We might consider looking at how the operational U.S. Air Force excels in addressing change and ask why we cannot employ those lessons in our own domain.

A case in point is the deployment of a new software-intensive system into a wing command post. All too often, from our perspective, it appears as if the task of transitioning new technologies and mission capabilities into operational use ends with the physical delivery of the system. (We have this image of a C-130 low-altitude parachute drop late at night, and then learn that the system has been deployed the following morning on CNN Headline News!)

We know this is not the method used to deliver a new aircraft or other more traditional weapon systems. From numerous field tests to tremendous amounts of operational training, the operators, air crews, and commanders for these new weapon systems are fully prepared for live combat before anyone would say their systems have been fully and effectively deployed. Why cannot the deployment of new software-intensive systems into wing command post operations follow the same proven process?

"We have wonderfully effective teams, yet many leaders and team members fail to appreciate what it takes to form a truly high-performance team that can sustain consistent execution at what often appears to be superhuman levels."

From our perspective, the way the Air Force deploys troops and assets for combat is close to perfection. Rather than focus on the differences, focus on what we can learn and what we might adapt to improve how we deploy mission-critical software-intensive systems into the command post.

How Does the Air Force Deploy for Combat?

At the heart of the Air Force's capability to deploy for combat is a standard process that serves in training as well as in real-world situations. This process demands realism and skills development in the context of their use. Backing up this realism is independent and objective observation, which provides real-time performance feedback and reinforcement. When people perform as instructed, opportunities become available. When performance differs from what others expect, there are consequences: more training and evaluation. Repeated shortfalls result in career consequences.

The Air Force cannot afford to have *creative* solutions to standard problems without careful coordination. Costly lessons we have learned about the *fog of war* necessitate that everyone must operate on the trust and faith that everyone else is doing what they are expected to do. Uncoordinated *creative* solution deviations often mean equipment and, more importantly, people, are not where they are supposed to be. The consequences of this in actual combat are dire.

The list of critical components for the Air Force's deployment process begins with the individual. Is the individual properly trained and equipped with the tools to perform the mission? Has the individual completed all legal and other family preparations so these concerns do not become awkward distractions? Have all medical precautions been honored so that the warrior is truly ready for worldwide duty?

With a properly prepared group of individuals, the team becomes the next focus. Teams train for deployment together, they go through the pre-deployment process together, and they deploy to combat together. Upon arrival, they will live together. The training makes the situation seem

familiar, and the presence of teammates reinforces that familiarity. Everyone supports one another to honor their roles and responsibilities, just as they were trained.

Prior to actual combat operations, the current rules of engagement are reviewed and made personal. What actions are permissible, and what implementation options are there? Backing up the rules of engagement is the commander's intent, which is a description of the purpose of the operation and its intended end state. The commander's intent helps to explain what the commander wishes to occur, so that everyone is equipped with additional information to help fill in the blanks. Oftentimes, reality provides a situation well outside of the bounds imagined by those who created the explicit guidance or the rules of engagement. Commander's intent is key to helping personnel to navigate these situations in which some or all of the rules may no longer apply, or no longer provide sufficient guidance.

The moment the transport stops moving at the deployment point, the set-up phase begins. From securing the site from hostile action to the more mundane acts of providing for water, food, latrines, and places to sleep, everyone plays a role and takes part honoring many months of practice.

From a secure and properly established base of operations, warriors fight and learn from each engagement. How can the mission be performed so that the warriors and resources are available to ensure not just a successful operation today, but tomorrow and the next day, until the conflict is over?

A recurring issue is the shifting list of unknowns and the often more troubling unforeseen unknowns. Survival depends on recognizing new patterns and appreciating that proven solutions may become obsolete at the most inopportune times. Knowing when to stop *Plan A* and resort to *Plan B* is not something that can be easily taught, yet it is a skill that must be mastered.

How We Should Deploy Technology for Operational Use

The successful principles used to train combat warriors are not being used to train information technology (IT) personnel to be successful in using new software-intensive systems being deployed. It is natural to assume that a console operator with years of experience should be able to figure out the features of a new system, yet no one would be so cavalier when it comes to aircraft and pilots. Certifications and qualification on one type of aircraft mean little when it comes to a new type of aircraft. If the proper certifications and qualification for a particular aircraft have not been earned, the warrior is not combat-ready in this new aircraft. Given the critical role that many software-intensive systems play in our operations, can we really believe that mistakes in this domain are not as devastating?

Since systems and their roles are so different, no single path to success is possible. Therefore, we should interpret the success of the Air Force combat-deployment process into a series of questions for those who are charged with deploying software-intensive systems into the operational Air Force. If we follow the process for combat deployment, the deployment of a new software-intensive system begins with the individual as follows:

- Are all individuals properly trained, certified, qualified, and skilled to the degree that the individual is predictably able to perform the new roles and honor the new responsibilities demanded by the new system? Is this training at the same breadth and depth as typically employed with more traditional weapon systems?
- Has the individual been trained in a team setting with the same team likely to use the system?
- Has the team had enough realistic training, performance evaluation, and feedback to ensure mission capability?
- Have clear rules of engagement been defined so it is clear to each and every member of the team (and those who lead them) what is to be done, how to do it, and how to employ the commander's intent when issues fall outside the scope of the planned scenario?
- Does everyone have a clear understanding of the following:
  - The new system's role.
  - What kind of information does it provide?
  - Who needs the information?
  - The key interfaces and outputs for the system.
  - Has the new system changed the nature of the work being performed?
  - Has everyone participated in the activities to put everything together properly and ensure it is working as it worked in training?
- Has everyone who must work seamlessly together moved from the old way of thinking and acting during predeployment to the new way of thinking and acting now that the system is about to go operational?
- Once the system goes operational, how do we ensure that we are honoring our training, the rules of engagement, and the commander's intent?
- How do we distinguish between the normal jitters and the real performance problems of our people on the new technology? (There are a lot of warriors out there who are able to provide solid advice on this.)

Experience From a Command Post

We offer the following real-world experiences as an example of lessons learned about the efficient deployment of software-intensive mission systems. While the example is simple and the solution might appear obvious, the fact remains that it was a serious operational problem that required significant time and resources to correct.

A wing command post has access to a wing commander and his or her staff at any time and any place, day or night. This is a unique role given to no other unit, and carries with it tremendous responsibility and mission-critical consequences.

Accurate and efficient notification to wing leadership is one of the primary responsibilities of the command post. In order to achieve this, three things are required: accurate and complete information concerning the situation, efficient and timely contact with wing leadership, and effective communication of pertinent information.

One of the authors was a command post chief whose early days were riddled with complaints that his troops were not executing their notification duties very well.

Upon investigation, the three requirements listed above were assessed. Collecting accurate and complete information about situations was not the problem; analysis of the complaints verified this. There were checklists covering almost every conceivable situation, and the author's controllers had most of them memorized by heart. They could instantly locate the appropriate checklist, and documentation showed that they accurately recorded all the pertinent information related to each event.

Achieving efficient and timely contact with the appropriate leader was not the problem either. The command post has a large screen on the wall that displays all contact information. Every wing leader is listed along with their current location, current contact phone number, and every other means of contact available. As leaders go about their activities, they update their contact information and location each and every time it changes. Change logs and related documentation showed no problem

here. An assessment revealed that the problem was in the effective communication of pertinent information between command post controllers and senior wing leaders.

Some context is required to fully appreciate the scope of this problem and the complexity of solving it. Command posts are generally staffed with young, inexperienced, and relatively lower-ranking individuals. They are still learning what the Air Force is all about and often do not even realize the criticality of their duties. Because of their age and limited experience, most controllers lack confidence in their abilities to interact with senior leadership. When you consider that their primary *customers* are very experienced field-grade officers in leadership positions, it is easy to understand the pressure and intimidation they feel when trying to communicate mission-critical information in pressure situations.

The study showed that, under pressure, young and inexperienced controllers would often make incomplete and inaccurate statements. It characterized the problem as one of *frame of reference*. Due to a lack of knowledge and lack of experience, there was no way for these young controllers to know what information was most important for the leader to know. The controller, having no insight to the questions the leader needed to answer in order to take appropriate action, was not able to wade through the data already collected in order to highlight the truly important facts. Having never been in a wing leadership position, the controller did not have the necessary experience to craft a proper message to support the leader's thought process about what action to take in response to the situation. To make a bad situation worse, these incidents always seem to occur at 3 a.m. when the general is sleeping and the spouse answers the phone!

A review of the training program, employing the checklist above, revealed shortfalls that led to the current performance issues. The training program focused on what information to collect, and on ensuring it was accurately recorded. Controllers were well trained to make efficient and timely contact and were provided with the necessary tools to accomplish this. When it came to effective communication of pertinent information, however, training failed to give the controllers the skills and confidence they would need to perform adequately. While there was scenario training that taught the proper identification of checklists, collection of pertinent data, and taking proper actions, controllers were never required to practice their communication skills or never given specific experiences to develop knowledge and insight that would enable them to craft more value-added messages.

The solution was to modify the training to provide more mission knowledge and insight, and to emphasize the development of their real-time skills. Scenario training was expanded to include mock phone calls with, among other things, a belligerent and sleepy general and a list of likely concerns about the event. A confidence-building component was also added to the training that served to empower the controllers to be more than just a relay mechanism. Their role is not just to pass on information; rather, it is to assist the leader to take the appropriate action in a timely manner. Depending on the situation, the controller is often more knowledgeable about the proper course of action than the leader they are talking to.

Controllers are experts in emergency actions and responses whereas wing leaders generally are not. In these situations, it is imperative that the controller be able to assume the proper role of an advisor and make specific recommendations to the senior leader. Without the confidence that they were an active and contributing component of the process and fearful of making technical mistakes with the data, our controllers were unable to perform their critical role effectively.

The results from the changes to the training program were outstanding. Controllers gained the confidence necessary to not only execute their duties more effectively, but many also became quite comfortable speaking with the wing leadership. Seeing the new confidence from command post controllers, wing leaders softened their demeanor, which further removed pressure leading to even better performance. In fact, many wing leaders began to know most of the controllers on a first-name basis. This was certainly a confidence and morale boost for the troops as they began to receive informal recognition for their efforts and improved performance.

It is common for warriors in IT jobs to be trained in the technical aspects of the job (e.g., the equipment, the data, the procedures, and the checklists) and not in how these systems are being used by others to provide mission value. The application of scenario training that more realistically reflected real-world situations seems obvious, but it was not obvious to the people who developed the previous training. The simple fact is that the controllers had not been trained under the same conditions and circumstances in which they would actually do the work.

It is also common for people to underestimate the amount of information an event might generate and to ignore the potential benefit derived from reordering information. Armed with the context provided by commander's intent, properly trained and skilled controllers become a powerful data processing component in a complex system. Their understanding and insights can shave crucial seconds from the critical path to an informed decision.

Lastly, it is important to understand who is on the team, the role each person plays, and how to flawlessly interact with each and every one on the team. Excluding the leaders from our definition of team was a mistake. Since the whole team did not train together in realistic situations, we did not have the insights needed to build the trust and confidence necessary to operate at the required performance level. When we applied standard Air Force warrior training concepts to jobs, the same kinds of benefits were achieved.

Conclusion

It is so easy to get consumed with the technical challenges that we forget the painful lessons we have learned over and over again. We in the Air Force have learned that everyone, under pressure, will fall back to behaviors that may not align with current mission needs.

In the early days of aviation, the loss of too many aircraft due to wheel-up landings resulted in creating a specific landing gear checklist item and performing drills upon drills to always use the checklist. The Air Force realized that experienced pilots, under pressure and interrupted in the landing process, were unable to do a perfect context-switch from landing to interruption and then back to landing as required in order to remember which landing stage they were in. Was it the last landing where the gear had already been lowered, or was it this landing where they had not?

The solution to the wheels-up landing problem was not more training about technical aspects of landing and the role of the gear. The solution was to recognize the important role humans play in the total system, to understand their limitations, and to develop mechanisms to help these dedicated warriors to act more perfectly than most would believe is humanly possible.

If our software-intensive systems are critical for mission success, we should leverage proven Air Force methods to ensure everyone performs as required, be it a combat deployment or an IT system deployment.◆

About the Authors

Scott B. Dufaud is a senior program manager for Software Engineering Services, Inc. based in Omaha, Neb. He is a retired Lt. Col. from the U.S. Air Force. During his military career, Dufaud was the director for the U.S. Air Force Software Process Improvement and deputy director for the U.S. Air Force Year 2000 Program Management Office. Since retirement, he has worked as a senior member of the technical staff at the Software Engineering Institute and as principal management consultant for SERENA Software Inc.

Software Engineering Services, Inc.
1508 JFKennedy Drive, Suite 201
Bellevue, NE 68005
Phone: (800) 244-1278
Fax: (402) 292-3271
E-mail: sdufaud@sesc-us.com

Lynn Robert Carter, Ph.D., is a principal fellow at the Institute for Software Research International at Carnegie Mellon University (CMU) where he has been a senior researcher and educator for nearly 14 years. At the Software Engineering Institute, his work included software technology adoption support to numerous Air Force, Navy, Marine Corp, and commercial customers. He is currently supporting the development and deployment of professional master's degree programs at CMU's West Coast campus at Moffett Field, Calif. Prior to CMU, Carter developed software at commercial firms for 17 years.

E-mail: lrc@sei.cmu.edu

BackTalk Answers

1. The Hundred Years' War lasted 116 years, from 1337 to 1453.
2. Panama hats traditionally come from Ecuador.
3. Sheep (sometimes horses) give us catgut.
4. The October Revolution is celebrated in November.
5. Squirrels give us camelhair brushes.
6. The Canary Islands are named for the large number of dogs there. Remember that the Latin word for dog is canis. The original name for the island was Insularia Canaria: Island of the Dogs.
7. Queen Victoria expressed a desire that no future king be called Albert. Thus, in 1936, when Albert ascended the throne, he respected her wish and took the name King George VI.
8. It's a bright crimson.
9. The tasty berries are from New Zealand.
10. Well, why else would it be named the Thirty Years' War? Actually, there were two Thirty Years' Wars, one from 1618-1648, and another from 1733-1763. The second war consisted of three wars: the War of Polish Succession, the War of Austrian Succession, and the Seven Years' War. The War of Polish Succession was not fought in Poland, but in Belgium, Lorraine, Lombardy, Naples, and Sicily. The Seven Years' War, of course, lasted seven years[1].
11. Gotcha! He was the son of a Roman official and his British wife. He was Welsh. He ended up in Ireland after being kidnapped at the age of 16 and taken there by Irish pirates[2].
12. Comes from China, of course.
13. Well, she was born in April; April 21, 1926 to be exact. However, in Britain her birthday is officially celebrated on the third Saturday of June. In Canada, the Queen's birthday is celebrated on the Victoria Day holiday in May. In Australia, the Queen's birthday will be celebrated this year on Monday, June 9, 2003, with the exception of Western Australia, which will celebrate on Monday, Sept. 29, 2003[3].
14. Your call.
15. Well, OF COURSE!

Notes

1. The Cave Online. "The Thirty Years' Wars." <www.thecaveonline.com/APEH/thirtyyearswar.html>.
2. Bryson, Bill. The Mother Tongue – English and How It Got That Way. Perennial Press, 2001: 49 Footnote.
3. "Birthday of Queen Elizabeth II, the Queen of Australia" <www.statusquo.org/birthday.htm> and "Frequently Asked Questions About Queen Elizabeth II" <www.mun.ca/library/ref/qeiifaq.html>.

Everybody Knows It's True!

Before you read this column, please take the following short quiz[1]. In the 15 statements listed below, circle any that are false (see answers on page 34).

1. The Hundred Years' War lasted 100 years.
2. Panama hats come from Panama.
3. Catgut comes from cats.
4. Russians celebrate the October Revolution in October.
5. A camelhair brush is made from camels.
6. The Canary Islands are named after the large number of canaries found there.
7. The first name of King George VI was George.
8. A purple finch is purple.
9. Chinese gooseberries come from China.
10. The Thirty Years' War lasted 30 years.
11. Saint Patrick was Irish.
12. India Ink comes from India.
13. Queen Elizabeth, who was born in April, celebrates her birthday on Queen's Birthday in April.
14. My metrics tell me the status of my software projects.
15. The Software Technology Conference is the best place to find new ideas to improve your day-to-day software efforts.

How well do you think you did? Since we're all software developers, I know you have already checked your score. In fact, you probably checked out each question as soon as you read it, didn't you?

Then you know it seems that a lot of commonly accepted things are not exactly so, huh? One of the lessons that I learned many years ago is that an array named Sorted_Data might not actually contain data that is sorted. Calling it sorted doesn't actually make it so – you have to examine the code yourself and verify that it is sorted correctly. Likewise, just having a file named Completed SCRs doesn't really mean that the software change requests entered there have actually been completed, does it? There has to be some verification and validation process to make sure they are really completed.

Here's a current example. I'm writing this column while at the annual SEPG conference in Boston. Scanning the headlines in the local paper, I read about the project known as the "Big Dig." To quote the Web site from the History Channel[2]:

> Boston's Big Dig is the most complex and expensive highway project ever undertaken in the United States. The city is replacing an outdated highway infrastructure with a new state-of-the-art highway system, most of which will be underground or underwater.

The 11-year project was scheduled to cost $10.8 billion, but the current budget is now $14.6 billion. The *Boston Globe* ran an article titled "Group Questions Big Dig Savings[3]." Imagine that – there have been cost overruns and schedule delays. First, an oversight committee said that the Big Dig schedules are unrealistic, promising opening dates for the tunnels that continually slip. It said that "failures in coordination" have lead to "persistent scheduling problems." Also, it concluded that Big Dig managers do not undertake "sufficient analysis" to justify expensive work acceleration to meet schedules.

There were also comments on questionable cost savings, too much management, and a need for independent peer review. In addition, the oversight committee disputed the project's claim of saving more than $1 billion by "fast-tracking" (i.e., starting actual construction before the design is completed). In their report, they said, "The committee does not believe that the full cost impact of changes, claims, and delays from incomplete coordination of work packages has been fully considered."

The private sector managers, of course, dispute some of the oversight committees' findings.

You know, this story is just like almost every software project that I've ever participated in! Why did I find this project so interesting? Because just the other day, a workshop student asked me, "Whose metrics do I trust? There always seem to be multiple interpretations of what the metrics mean, and I don't know whom to believe!"

He hit the heart of the problem – what do the numbers mean? You see, a measurement (which is a piece of data) doesn't indicate anything; it's only a number. Put enough measurements together though, and you get a metric – a trend analysis that gives you indicators of how you are doing – whether you are doing better or worse than you were doing before.

But what do they mean? Unfortunately, that's where the problem lies. It's the interpretation of the metrics that causes problems. We don't really know what is good or bad; we only know if the numbers are higher or lower than before.

I'm already out of space, and I've barely covered the problem. Remember that numbers are just numbers. The meaning you attribute to them is a result of analysis. Do you trust the numbers? Do you trust the analysis? Just because a number says you are doing well, remember that the numbers are not reality but only somebody's reflection and interpretation of reality. In short, it's not necessarily true just because everybody says so!

–David A. Cook
Software Technology Support Center/
Shim Enterprises, Inc.
david.cook@hill.af.mil

Notes

1. I wish I could take credit for this quiz. Various versions have been floating around on the Web forever. So, no authorship claimed. (OK, I added questions 11, 12, 13, 14 and 15.)
2. HistoryChannel.com <www.historychannel.com/cgi-bin/frameit.cgi?p=http%3A//www.historychannel.com/exhibits/bigdig>.
3. Raphael Lewis. "Group Questions Big Dig Savings." The Boston Globe 22 Feb. 2003: B1.

We Are Your Partner for Dynamic Solutions

Aligning software technologies and processes with your organization's strategy, infrastructure, and personnel gives you an advantage in today's environment of tight budgets, information overload, and changing customer requirements.

We provide knowledge, experience, and results for government organizations in:

- Capability Maturity Model® and Capability Maturity Model Integration℠
- Configuration Management
- Documentation and Standards
- Independent Expert Program Reviews
- Independent Verification and Validation
- Interim Profiles and CMM® Appraisals
- Object-Oriented Adoption, Transition, and Migration
- Personal Software Process℠ or Team Software Process℠
- Process Definition
- Programming Language Support
- Project Management
- Requirements Engineering and Management
- Risk Management
- Software Acquisition
- Software Cost Estimation
- Software Measurement
- Software Process Improvement
- Software Quality, Inspection, and Test
- Strategic and Action Planning
- Theory of Constraints

CROSSTALK

STC

Software Technology Support Center

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

Sponsored by the Computer Resources Support Improvement Program (CRSIP)

Published by the Software Technology Support Center (STSC)

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

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