CSC
NORTH AMERICAN
PUBLIC SECTOR

Software Sustainability – Challenges for Acquisition, Engineering, and Capability Delivery in the Face of the Growing Cyber Threat

Paul R. Croll
Fellow
CSC
pcroll@csc.com

Outline

- Maintenance vs. Sustainment
- Software Sustainability Issues
- Sustainability and Cybersecurity
- Challenges for Acquisition, Engineering, and Capability Delivery

CSC

NORTH AMERICAN
PUBLIC SECTOR

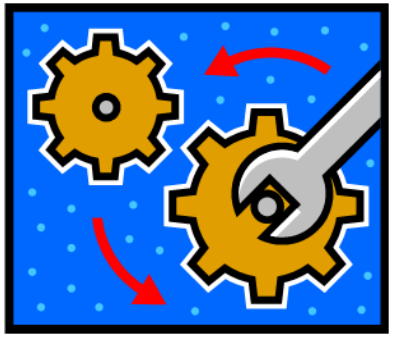
Maintenance vs. Sustainment

Maintenance vs. Sustainment

- ISO/IEC/IEEE 14764 [1] describes four classes of *software maintenance*:
  - Corrective
    - Modifications necessitated by actual errors in a software product
  - Preventive
    - Detection and correction of latent faults in a software product after delivery
  - Adaptive
    - Modifications necessary to accommodate a changing environment
  - Perfective
    - Modifications that provide new functionality or improve the software product's performance or maintainability
- DoD 5000.2 [2] describes *sustainment* as including:
  - Supply; maintenance; transportation; sustaining engineering; data management; configuration management; HSI; environment, safety (including explosives safety), and occupational health; protection of critical program information and anti-tamper provisions; supportability; and interoperability.

Software Sustainment

- Schmidt [3] suggests similar categories for software sustainment.
  - Corrective Sustainment
    - Diagnoses and corrects software errors after release.
  - Perfective Sustainment
    - Upgrades existing software to support new capabilities and functionality
  - Adaptive Sustainment
    - Modifies software to interface with changing environments
  - Preventive Sustainment
    - Modifies software to improve future maintainability or reliability.

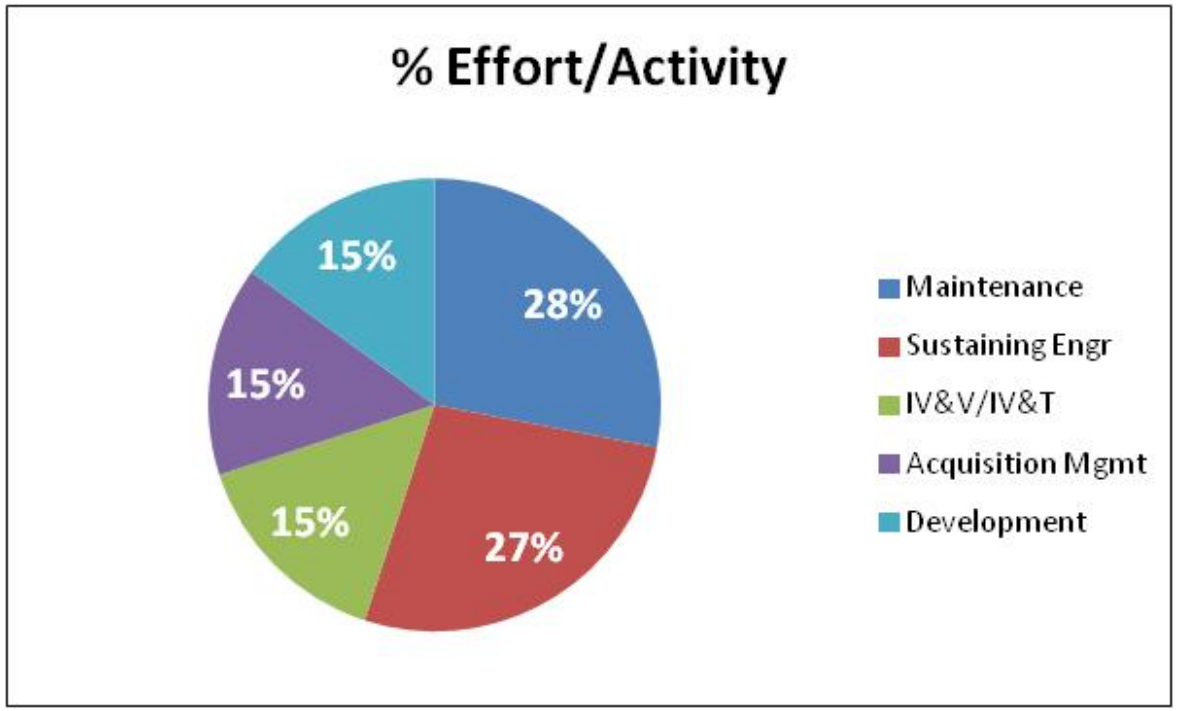


Software Sustainment Scope and Context

- Schmidt [4] defines the scope of software sustainment as:
  - Coordinating the processes, procedures, people, information, and databases required to support, maintain, and operate software-reliant aspects of DoD systems
- Schmidt also describes several software sustainment drivers including:
  - Rapid advances in hardware capability, which accelerates technology refresh cycles
  - The reliance on COTS and associated COTS integration practices, which impacts software engineering development skills and expertise in older technologies
  - Diminishing manufacturing sources considerations for both hardware and software
  - Budget constraints regarding modernization and recapitalization of legacy systems
  - Adapting existing systems to meet new threats and increased requirements for interoperability

Scope of the Sustainment Problem

- Sustainment will account for somewhere between 60 and 90 percent of the total lifetime costs of a software product [5]
- A study of Army and Air Force Maintenance Centers showed that over 50% of their software-related effort was spent on Maintenance and Sustaining Engineering [6]



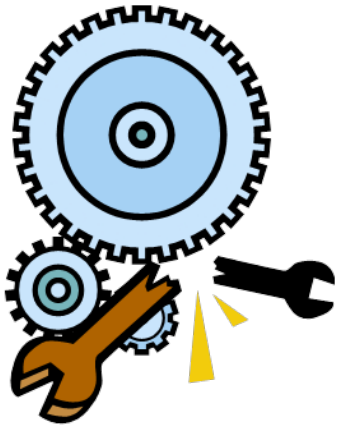
CSC

NORTH AMERICAN
PUBLIC SECTOR

Software Sustainability Issues

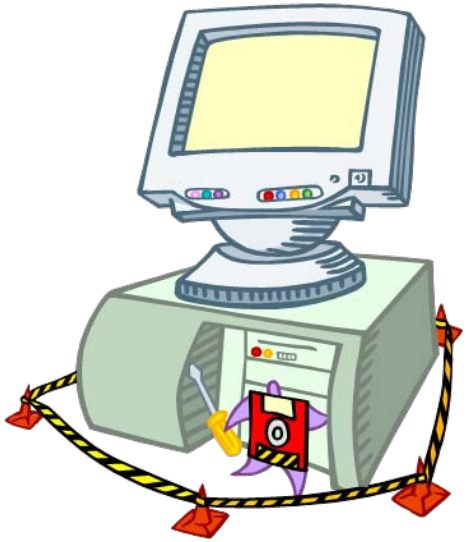
We have a crisis in software when we examine its use, reuse, and support. Sustainability in this context isn't just a funding issue; it's a matter of having comprehensible, extensible, reliable software to meet current needs. [7]

Jennifer M. Schopf
Office of CyberInfrastructure, National Science Foundation
CyberInformatics Group, Woods Hole Oceanographic Institute



Software Sustainability Issues

- Some typical software sustainability issues include [8]:
  - Lack of requirements traceability
  - Evolution of software versions or releases that are difficult or impossible to trace
    - Change history not documented
  - Unavailability of the original software development toolset
    - Compilers, linkers, loaders, etc.
  - Impossible to understand code
    - Large, complex systems
    - Systems with a long history of revisions
  - Documentation that is nonexistent, incomplete, or of poor quality
    - Must be consistent with the source code
  - Designs that do not easily support change
    - Architectures designed for supportability and extensibility



Additional Software Sustainability Issues

- Transition planning
- Maintenance environment and support database transition
- Documentation handoff
- COTS and Open Source license management
- COTS obsolescence and upgrade planning
- Code escrow
- Formal training for sustainers

Sustainability, Aging Systems, and Reuse

Size in FP	Civilian Government Projects	Military Projects	*Average*
1	12.00%	5.00%	*8.50%*
10	15.00%	7.00%	*11.00%*
100	20.00%	9.00%	*14.50%*
1,000	25.00%	12.00%	*18.50%*
10,000	60.00%	25.00%	*42.50%*
100,000	78.00%	30.00%	*54.00%*
1,000,000	88.00%	35.00%	*61.50%*
Average	*42.57%*	*17.57%*	*30.07%*

Figure 3. Estimated percent of Aging Software Applications with Structural Decay, Error-prone Modules, and Geriatric Problems: Capers Jones © 2008

Size in FP	Civilian Government Projects	Military Projects	*Average*
1	50.00%	72.50%	*61.25%*
10	45.00%	70.00%	*57.50%*
100	37.50%	62.50%	*50.00%*
1,000	22.50%	57.50%	*40.00%*
10,000	15.00%	45.00%	*30.00%*
100,000	12.50%	32.50%	*22.50%*
1,000,000	7.50%	30.00%	*18.75%*
Average	*27.14%*	*52.86%*	*40.00%*

Figure 4. Approximate percent of reused components and source code in U.S. Software Applications: Capers Jones © 2008

- For military projects, as one approaches systems the size of typical large combat systems (expressed as function points), the estimated percent of aging problems rises to 35%

Geriatric Problems and Reused Components Are a Potential Breeding Ground for Vulnerabilities

Software Entropy and the Total Cost of Ownership

- Jones [9] points out that entropy, the tendency of systems to destabilize and become more chaotic over time, has significant impact on the total cost of ownership for software
  - The implications of entropy for poorly structured, error-prone applications are:
    - Cumulative defect fixes and maintenance updates will degrade the original application structure to the point where the application eventually destabilizes
    - The application will eventually will become so complex that maintenance can only be performed effectively by a few experts who have long-term deep insight into the application
- Jones [9] also describes six major total cost of ownership elements:
  - The initial cost of building an application
  - The cost of enhancing the application with new features over its lifetime
  - The cost of repairing defects and bugs over the application's lifetime
  - The cost of customer support for fielding and responding to queries and customer-reported defects
  - The cost of periodic restructuring or refactoring of aging applications to reduce entropy and thereby reduce bad-fix injection rates
  - The cost of removing error-prone modules via surgical removal and redevelopment.

Sustainment Support Organization Issues

- Lapham and Woody [10] point out that how well the sustainment organization has prepared itself for handling COTS-specific maintenance issues has a direct bearing on the success of the sustainment effort
- Sustainment support organizations can exhibit the following deficiencies with respect to COTS-intensive systems [10]:
  - They assume that annual code refresh cycles are adequate
  - Their method of configuration management is primarily hardware-focused
  - There is a minimal, if any, vendor management plan to ensure key functionality remains available in the COTS products being used
  - They rarely use any formal evaluation criteria to determine or confirm when a system is ready to enter sustainment

The Impact of COTS-Intensive Systems on Sustainment

- Four critical areas provide unique challenges with respect to sustaining COTS-intensive software systems [10]
  - System obsolescence, technology refresh, and upgrade planning
  - Source code escrow
  - Vendor license management
  - Architecture and COTS software interfaces

System Obsolescence, Technology Refresh, and Upgrade Planning

- COTS iterations are marketplace driven
- A COTS management plan that addresses when and how updates and new releases will be incorporated is essential
- System evolution must address how all the COTS software components work together and how changes to one component can affect the others

Source Code Escrow

- Problems can arise when COTS vendors go out of business or cease to support a product
- If the customer does not have rights to the source code, sustainment of the system is impossible
- Escrow clauses in contracts require the vendor to place a copy of the code in escrow for release to the customer under defined circumstances
  - Source code alone may not be sufficient for sustainment. A copy of the development environment is often essential as well.

Vendor License Management

- The roles and responsibilities for transitioning software licenses should be clearly defined
- The budget for maintaining software licenses must be specified by the organization designated with maintaining the licenses.

Architecture and COTS Software Interfaces

- Sometimes, system-specific tailored versions of COTS products may be incorporated into a system
  - Over time, the customized version can significantly diverge from the supported product
- The system architecture should be designed such that the system as a whole is insulated from COTS product interfaces
  - When the COTS product changes, there should not be a ripple effect of changes to the interfaces
  - In addition, consideration needs to be given to COTS products that perform the same or similar functions so that options are identified for substitution
- The sustainment organization should participate in vendor users groups
  - Participation provides insight into the vendor's product roadmap
  - Early warning about future product changes that could impact the system

NORTH AMERICAN
PUBLIC SECTOR

Sustainability and Cybersecurity

Scope of The Cybersecurity Problem

Size in FP	Civilian Government Projects	Military Projects	*Average*
1	1	1	*1*
10	5	4	*5*
100	29	14	*24*
1,000	155	55	*120*
10,000	832	209	*600*
100,000	4,467	794	*3,031*
1,000,000	23,988	3,020	*15,412*
Average	*4,211*	*585*	*2,742*

Figure 1. Estimated Number of Security Vulnerabilities in Software Applications. Source: Capers Jones © 2008

Size in FP	Civilian Government Projects	Military Projects	*Average*
1	25.00%	5.00%	*11.29%*
10	35.00%	15.00%	*26.00%*
100	45.00%	20.00%	*33.57%*
1,000	62.00%	30.00%	*54.57%*
10,000	80.00%	35.00%	*74.00%*
100,000	87.00%	40.00%	*80.14%*
1,000,000	92.00%	45.00%	*86.29%*
Average	*60.86%*	*27.14%*	*52.27%*

Figure 2. Probability of Serious Security Vulnerabilities in Software Applications. Source: Capers Jones © 2008

- For military projects, as one approaches systems the size of typical large combat systems (expressed as function points), the estimated number of security vulnerabilities rises to above 3000 and the probability of serious vulnerabilities rises to over 45%
- The statistics are much worse for civilian systems. As we move more and more into COTS and open source software for our combat systems, one might expect that the true extent of vulnerabilities in our systems would lie somewhere between those of military and civilian systems.

COTS and Open Source Exacerbate the Problem

- Reifer and Bryant [11] studied 100 packages were selected at random from 50 public Open-Source, COTS, and GOTS libraries
  - Spanned a full range of applications and sites like SourceForge
  - Over 30% of Open Source and GOTS (Government Off the Shelf) packages analyzed had dead code
  - Over 20% of the Open Source, COTS, and GOTS packages had suspected malware
  - Over 30% of the COTS packages analyzed had behavioral problems
- Reifer and Bryant conclude that the potential for malicious code in applications software is large as more and more packages are used in developing a system.

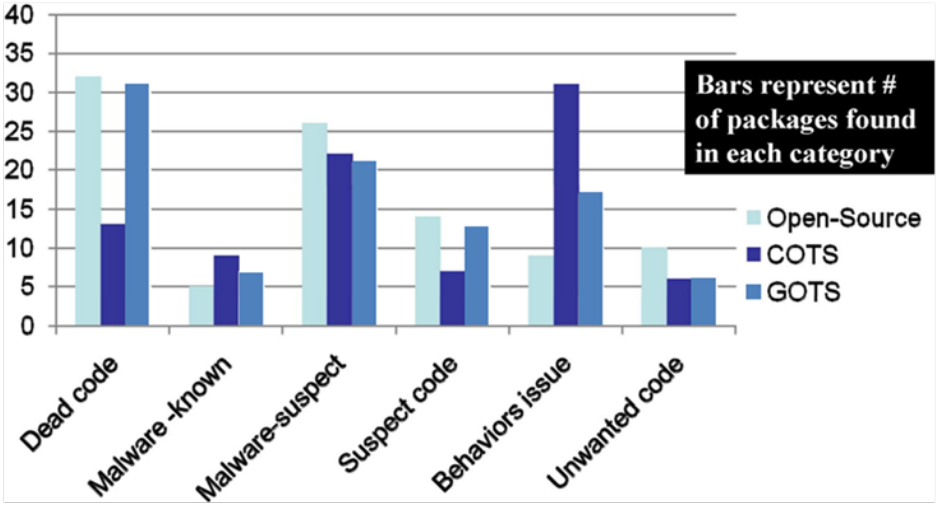


Figure 5. COTS Study Findings. *Source: D. Reifer and E. Bryant, Software Assurance in COTS and Open Source Packages, DHS Software Assurance Forum, October 2008*

It Is Difficult to Verify the Security of COTS Products

- Miller [12] describes COTS products as black boxes to their customers
  - No means to review the code or the architecture
  - Veracity of security claims relies on the developers reputation, published security reports, and security forums
  - Vendors coding practices are largely unknown
  - COTS software is generic and does not typically address your specific operating environment, requiring careful configuration for secure operation
- Miller also points out that COTS software is generally a more attractive target than custom code

Information Assurance for COTS Sustainment

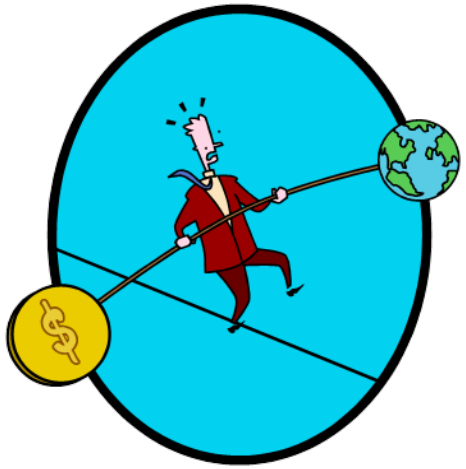
The transition to sustainment for systems containing COTS components must make special provisions to enable the sustainment organization to evaluate the impact of COTS component changes, independently assess the criticality of security flaws, and accommodate additional support for IA testing and validation. [13]

- Each COTS component may change at a different rate based on vendor support decisions
- The impact of each change must be evaluated against the mission that component performs in the fielded system
- The sustainment organization must acquire sufficient knowledge of the fielded system to evaluate the impact
- The sustainment organization cannot rely on a vendor to determine the criticality of a reported security flaw
- Skilled security experts familiar with the use of the component within the fielded system must evaluate each flaw based on the risk to the mission of the system and its potential impact

CSC

NORTH AMERICAN
PUBLIC SECTOR

Challenges for Acquisition, Engineering, and Capability Delivery



Challenges for Acquisition, Engineering, and Capability Delivery

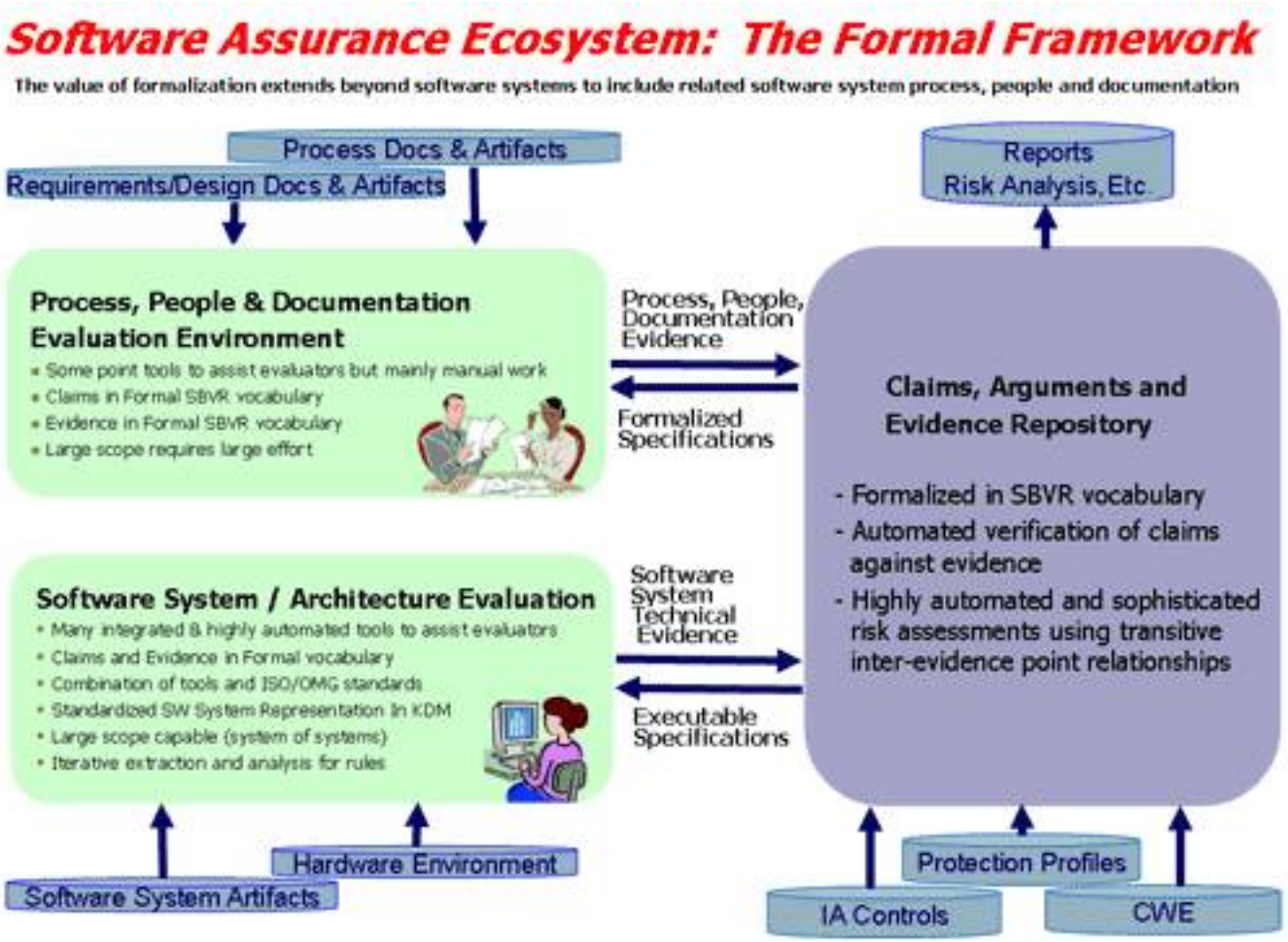
- Acquisition
  - Policy
  - Strategy
  - Total Cost of Ownership
- Engineering
  - Sustainable Architectures
  - Resiliency of New and Legacy Systems
  - Implications of Agile Development
- Capability Delivery
  - Characterizing Legacy Systems
  - Forward-Fit and Back-Fit Strategies for Sustaining Capability in the Changing Battle Space
  - Optimizing Capability Through Value Engineering

The Software Assurance Ecosystem as a Tool for Sustainability

- The SwA Ecosystem [14] is a formal framework for analysis and exchange of information related to software security and trustworthiness that:

– Provides a technical environment where formalized claims, arguments, and evidence can be brought together with formalized and abstracted software system representations to support high automation and high fidelity analysis

– Is based entirely on international (ISO/IEC/OMG) Open Standards
  - Semantics of Business Vocabulary and Rules (SBVR)
  - Knowledge Discovery Meta-model (KDM)
  - Software Assurance Meta-model (SAM) — work in progress for Assurance Case
    – Software Assurance Evidence Meta-model
    – Software Assurance Claims & Arguments Meta-model



References

[1] ISO/IEC/IEEE 14764:2006, Software Engineering — Software Life Cycle Processes — Maintenance. ISO, Second Edition. Geneva Switzerland, September 2006.

[2] U.S, Department of Defense, DoDI 5000.2, Operation of the Defense Acquisition System Enclosure 2 – Procedures, *8. Operations and Support Phase*, December 2008.

[3] D. Schmidt. The Growing Importance of Sustaining Software for the DoD, Part 2 SEI R&D Activities Related to Sustaining Software for the DoD, *SEI Blog*, Software Engineering Institute, Carnegie Mellon University, August 15,2011.

[4] D. Schmidt. The Growing Importance of Sustaining Software for the DoD, Part 1 Software Sustainment Trends and Challenges, *SEI Blog*, Software Engineering Institute, Carnegie Mellon University, August 1,2011.

[5] U.S. Air Force, Software Technology Support Center, Guidelines for Successful Acquisition and Management of Software-Intensive Systems: Weapon Systems Command and Control Systems Management Information Systems, Condensed Version, *Chapter 16, Sustainment and Product Improvement,* February 2003

[6] U.S. Army, Army Software Operations, Maintenance and Sustainment Study Overview, *Software Sustainment Collaborators Workshop*, September 14, 2011.

[7] Schopf, J. Sustainability and the Office of CyberInfrastructure. *Eighth IEEE International Symposium on Network Computing and Applications*, 2009.

[8] U.S. Air Force, Software Technology Support Center, Guidelines for Successful Acquisition and Management of Software-Intensive Systems: Weapon Systems Command and Control Systems Management Information Systems, Version 3.0, *Part 2: Engineering, Chapter 12, Software Support,* May 2000.

[9] Jones, C. Geriatric Issues of Aging Software, *Crosstalk, The Journal of Defense Software Engineering, Vol. 20, No. 12,* USAF Software Technology Support Center, Hill AFB, UT, December 2007.

[10] Lapham, M. and Woody, C. Sustaining Software-Intensive Systems, CMU/SEI-2006-TN-007, Carnegie Mellon University, May 2006.

[11] Reifer, D, and Bryant, E. Software Assurance in COTS and Open Source Packages, Proceedings of the DHS Software Assurance Forum, October 14-16, 2008.

[12] Miller, C. *Security Considerations in Managing COTS Software*, Cigital, Inc. 2006, Department of Homeland Security, Build Security In.

[13] Woody, C., GUIDELINE 8: Information Assurance for COTS Sustainment, in Novak, W (Ed.), *Software Acquisition Planning Guidelines*, CMU/SEI-2005-HB-006, Carnegie Mellon University, December 2005.

]14] Department of Homeland Security, *Software Assurance Ecosystem*, DHS National Cybersecurity Division, 2011. https://buildsecurityin.us-cert.gov/swa/ecosystem.html

For More Information . . .

Paul R. Croll
CSC
10721 Combs Drive
King George, VA 22485-5824

Phone: +1 540.644.6224

Fax: +1 540.663.0276

e-mail: pcroll@csc.com