Proceedings of the 10th Colloquium for Information Systems Security Education
University of Maryland, University College
Adelphi, MD June 5-8, 2006

Best Software Assurance Practices in Acquisition of Trusted Systems

Mary L. Polydys & Daniel J. Ryan, *National Defense University;* Julie J. C. H. Ryan*, George Washington University*

Abstract – Systems software and application software make it possible for our systems and networks to function effectively and efficiently, enabling creation, processing, storage and communication of the information assets that drive our economy and our way of life. Our dependency on the information infrastructure makes software assurance an essential element of national security and homeland defense. The interdependence of our critical infrastructures with the information infrastructure, the size and complexity of software systems, our increasing reliance on outsourcing for software development and maintenance, and the growing sophistication of malicious threats argue for increased rigor and use of software assurance methodology in developing or acquiring software. This paper provides a first look at some of the considerations shaping the Software Assurance Guide for Acquisition Managers, which will provide "best practices" and materials that can be used in acquisition and training and education.

Index terms – best practices; software assurance; software acquisition

I. Introduction

Cyberspace and the systems and networks that comprise its operational environment are the nervous system of those critical infrastructures that support the national and economic security of the nation. [1] Systems software and application software make it possible for our systems and networks to function effectively and efficiently, enabling creation, processing, storage and communication of the information assets that drive our economy and our way of life. Thus, our dependency on the information infrastructure makes software assurance an essential element of national security and homeland defense. In February, 2005, a report to the President of the United States [2] stated, "Vulnerabilities in software that are introduced by mistake or poor practices are a serious problem today. In the future, the Nation may face even more challenging problems as adversaries—both foreign and domestic—become increasingly sophisticated in their ability to insert malicious code into critical software."

Some experts believe that craftsmanship, focusing as it does on programming skills, is an appropriate model for encouraging responsible practices that lead to the development of trusted software with required levels of integrity. Freeman Dyson, in his essay "Science as a Craft Industry," says, "In spite of the rise of Microsoft and other giant producers, software remains in large part a craft industry. Because of the enormous variety of specialized applications, there will always be room for individuals to write software based on their unique knowledge. . . . The craft of writing software will not become obsolete." [3]

There is a distinct difference between simply writing software and writing structurally sound and competently engineered software. Some in the community question whether a focus on craftsmanship is sufficient to assure the integrity, security and reliability of software when we need to "reduce our vulnerability to debilitating attacks against our critical information infrastructures. . . ." [4] The interdependence of our critical infrastructures with the information infrastructure, the size and complexity of software systems, our increasing reliance on outsourcing for software development and maintenance, and the growing sophistication of malicious threats argue for increased rigor and use of "systematic, disciplined, quantifiable approach[es] to the development, operation, and maintenance of software" [5] – in short, software assurance. Properly applied, such an approach can "facilitate … best practices and methodologies that promote integrity, security, and reliability in software code development, including processes and procedures that diminish the possibilities of erroneous code, malicious code, or trap doors that could be introduced during development." [6] A critical challenge in this endeavor lies in the interfaces between software products, developed or acquired separately as either component

Mary Linda Polydys is Chair, Information Operations & Information Assurance Department, Information Resources Management College, National Defense University, Washington, DC, polydysm@ndu.edu; Daniel J. Ryan is Professor of Systems Management, Information Resources Management College, National Defense University, Washington, DC, ryand@ndu.edu; Julie J. C. H. Ryan is Assistant Professor, Engineering Management and Systems Engineering Department, School of Engineering and Applied Science, George Washington University, Washington, DC, jjchryan@gwu.edu

elements of larger systems or as stand-alone systems, that must work together without violating security goals.

In reaction to the need to both understand and conceptualize management answers to these challenges, software assurance is receiving high level attention. In 2003 the US Department of Defense (DoD) launched a Software Assurance Initiative led by Joe Jarzombek, then Deputy Director for Software Assurance, Information Assurance Directorate, Office of Assistant Secretary of Defense (Networks and Information Integration). This effort was joined in 2004 by the Department of Homeland Security (DHS) to address concerns of critical infrastructure vulnerabilities caused by poor-quality, unreliable, and susceptible software. In March 2005, Mr. Jarzombek became the Director for Software Assurance, National Cyber Security Division within DHS, retaining his leadership role in the collaborative interagency effort to incorporate software assurance into critical software projects.

As part of this effort, working groups, with members from government, industry, and academia, were established in 2004. The working groups focus on the topics of:

- Software Technology, Tools and Product Evaluation
- Software Acquisition
- Software Processes and Practices
- Workforce Educational and Training

All of the working groups are focused on the problems associated with software trustworthiness and security, but from different perspectives because this is such a large problem space. The Software Acquisition Working Group, whose efforts are the focus of this paper, is addressing how to leverage the acquisition process in order to ensure the safety, security, reliability and dependability of software. This Group was formalized and met on 28-29 November 2005 to begin development of a Software Assurance Guide for Acquisition Managers. The guide will provide "best practices" and materials that can be used in acquisition and training and education. The guide will include two critical tools for participants in software acquisition programs:

- Templates for acquisition language and evaluation based on successful models, and
- Common or sample statements of work/procurement language that includes provisions on liability for federal acquisition managements. [7]

These tools are intended to assist acquisition participants in setting standards, operating with best practices, and managing software acquisition with an eye towards meeting security requirements. This paper provides a first look at some of the considerations shaping that Guide.

II. Software Assurance

There is a non-trivial issue associated with language when talking about security. The word "trust" has special meaning to formalists in computer security and has additional special meaning to those who understand the role of a trusted kernel as a sort of security hall monitor in computer programs. In this document, the word "trust" is not used with that level of formalism and the reader's indulgence is begged. In this paper, the words "trust" and "secure" are used to imply a reasonable level of assurance that a software module has some resistance to vulnerabilities due to competent design and development in which potential problems were actively considered and designed to be mitigated. Note the use of "reasonable level of assurance" terminology – this is a reflection of the reality that it is very difficult to have an absolute level of assurance, particularly with very large systems. Hence, trusted software does what it is designed to do accurately and reliably, and does not do or allow anything other than that which was designed as explicitly allowable. In other words, there is a level of confidence that the software functions as intended and is free of both intentional and unintentional vulnerabilities, including those potentially engendered through interfaces with other systems.

Unfortunately, history has shown that if security is not a specific requirement in the acquisition or development phase, software may satisfy stringent functional specifications and yet be riddled with vulnerabilities. Technical factors, psychological factors and other economic and social factors make development and implementation of trusted software extremely difficult. [8] Some of the largest software companies in the world have brought software to market that was not only untrusted, it could not easily be made trusted because of an architecture that had been designed without security as a consideration. Richard Forno, in "Who Needs Hackers? We've got Microsoft" says, "Years of service packing, patching, re-patching, updating, critical updating, and hotfixing Windows products have made them dirty and prone to breaking, as we see every few months…. Microsoft needs to revisit the basic design of Windows - namely, removing the shared code between applications and the underlying Windows operating system. Actually, a worldwide recall might in order." [9]

In order to change both the climate of software development and the results of software acquisition, the concept of software assurance must be integrated wholly into the existing structures. Software assurance implies a rigorous process for acquiring, developing, deploying and operating trusted software based on "measures that [are]

employed to protect the systems from software vulnerabilities and unintended software processing that expose a system to compromises in availability, integrity, and other security properties." [10]

A US General Accounting Office report to the US Congress in May 2004 [11] emphasized the inherent vulnerability of defense software acquisition to problems resulting from inadequate security considerations during the acquisition process, and recommended to acquisition managers that they:

> Require program managers, working with software assurance experts, acquisition personnel, and other organizations as necessary to specifically define software security requirements, including those for identifying and managing software suppliers. These requirements should then be … used as part of the criteria to select software suppliers.
>
> Based on defined software security requirements, require program managers to collect and maintain information on software suppliers, including software from foreign suppliers. This information should be evaluated periodically to assess changes in the status of suppliers and adjustments to program security requirements. [12]

III. Secure Coding

Whether we decide to design, develop and implement software in-house, or contract out those tasks, creating trusted software requires that we write code that is designed with attacks in mind; robust code that is specifically designed to resist attack. The discipline that undertakes such development is called secure coding, although that name is somewhat of a misnomer since the discipline addresses much more than coding practices. [13]

If we perform the work in-house, we need system specifications that require incorporation of application security at every stage of the system's life cycle. Security is difficult or impossible to add into software as an afterthought. We need security that is "baked in," not "bolted on" unless we are prepared to perpetually operate in a system-high, closed facility environment with no network connectivity. If we outsource the development, similar requirements must be included as contractual requirements by acquisition managers. This brings the process of acquisition into sharp focus..

IV. Acquisition—Secure System Requirements

The acquisition of secure software is dependant on a set of quality system requirements. System requirements, properly written to include "baked-in" security, will ensure that

- security is a primary consideration for the system architecture, including interfaces to other systems,
- security is constantly considered during the design phase,
- security requirements are part of the "report card" for acquisition program managers so that these requirements are not unfunded mandates that are passed on to operational program managers,
- coding practices specifically developed to enable and enhance security are used during implementation,
- security is treated as critical during integration,
- test plans and procedures ensure that the software does what it is designed to do, and does not do anything other than what it was designed to do,
- the software is deployed and installed in a manner that preserves its security features,
- change management processes preserve system security, that system operations do not compromise security,
- the system is properly operated and maintained in a manner that preserves security,
- and, finally, at the end of the system's life cycle, the system is decommissioned and disposed of without loss or compromise of sensitive data or information.

The ultimate goal of software assurance during the acquisition process is the production, deployment and use of software that is trusted, predictable, and conforms to the security policy for the system. Trustworthiness implies that serious effort has gone into minimizing that exploitable vulnerabilities accidentally or purposefully exist in software. Predictability implies that there is high probability that the software functions accurately and reliably as intended. Conformance implies that there are a "planned and systematic set of multi-disciplinary activities that ensure software processes and products conform to requirements, and applicable standards or procedures." [14]

Note the recurring use of the terms reasonable and probability. It is not a secret that it is impossible to achieve total trustworthiness or security. As such, there is

a very large requirement for risk assessment, mitigation, and management in the acquisition process.

V. Acquisition—Risked-Based Process

All decisions in the acquisition process should be made based on not only costs and benefits, but also risks. This is particularly important when making decisions about the acquisition of secure software. Reasonable risk taking may appropriate as long as a risk assessment results in the determination that certain risks under certain conditions are acceptable. Acquisition of secure software must begin with a risk analysis. Risks must also be controlled and mitigated and residual risks understood. Therefore, risk management is essential throughout the process of acquiring secure software.

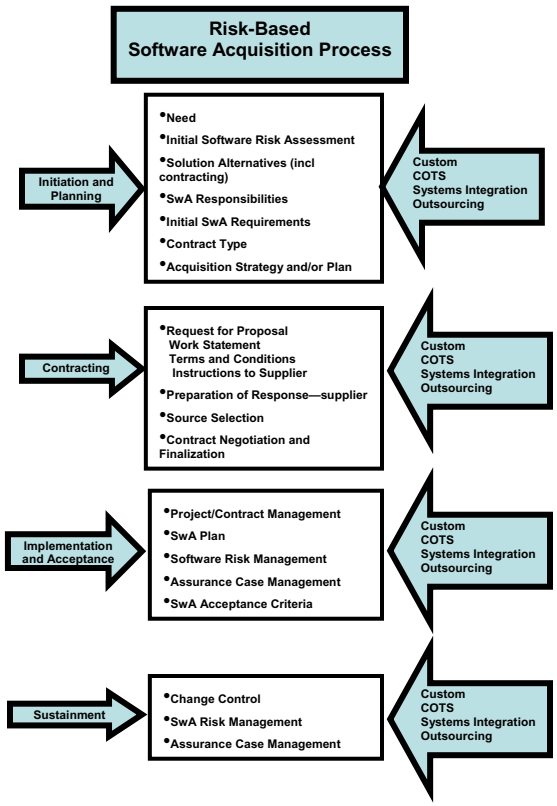

Figure 1. Risk Based Software Acquisition [15]

The many and varied specific techniques in a risk-based software acquisition process are being addressed by the working group, with specific (summarized here) recommendations that are allocated among the following phases:

- the Initiation and Planning phase incorporates an initial risk assessment,
- the Contracting phase requires the supplier to provide a risk management process,
- the Implementation phase requires risk management by both the acquirer and supplier, and
- the Sustainment phase requires the continuation of risk management.

The integration of these risk management elements phase requirements for software assurance are shown in Figure 1 and discussed in the following sections.

VI. Acquisition—Initiation and Planning

During the Initiation and Planning phase, a risk assessment provides the foundation for the architectural and design tradeoffs among alternatives, software and software assurance requirements, assignment of responsibilities, and acquisition strategy or plan. Specifically this phase includes determining the need for security, conducting the initial risk assessment, determining solution alternatives, creating security requirements, assigning responsibilities, and developing the strategy or plan.

Determining the need for security is, from a conceptual perspective, a foregone conclusion. However, the stringency of security requirements, the level of trustworthiness, and the level to which security measures must be proven to work correctly and not work incorrectly will vary from system to system. The nature, purpose, and architecture of the software to be acquired will dictate the level of security needed and the definition of "reasonable level of assurance." In other words, this analysis results in both a comprehensive statement of need for security unique to the function and operational environment of the software and also an assessment of how thoroughly security controls must be tested.

Conducting the initial software assurance risk assessment is done using a risk-based categorization scheme to facilitate software assurance risk identification and is useful in standardizing the results of assessing potential security risks for the software. Such a security categorization scheme is based on the software's criticality to the organization's mission, the sensitivity of the information that it processes, and the environment in which it is intended to operate. [16, 17]

Determining the solution alternatives is performed based on the initial risk assessment so that plausible alternatives can be identified. This must be done on a functional level that is implementation independent to avoid falling into the design traps associated with preformed solution judgments.

Creating initial software and software assurance requirements is done to identify those solutions whose risks are acceptable which are then translated into initial software requirements. The software assurance requirements are derived from the risk mitigation

strategies that must be employed to maintain that acceptable risk.

Determining software assurance responsibilities is imperative so that adequate resources for ensuring software assurance during the acquisition process are identified up front with an acknowledgement that resources equate to cost. Furthermore, this explicit identification links performance to specific roles so that there is accountability in the process.

Developing the acquisition strategy or plan is done last, so that the decisions regarding software assurance are captured in a strategy or plan that can then be used to guide the acquisition. [18]

After the Initiation and Planning phase, the project moves through Requirements Specification and Contracting, if an outsourcing strategy is chosen. As contracting is the most common way to acquire software, it is worth considering how the elements of software assurance play out during this phase.

VII. Acquisition--Contracting

In the contracting phase, controls must be in place to prevent the possibility of vulnerabilities being introduced into software. Vulnerable software may permit unintentional errors leading to faulty operations resulting in loss of life, destruction of information, or major disruption of operations; intentional insertion of malicious code intent on destruction of information, major disruption of operations, or even destruction of critical infrastructures; theft of classified information; theft of personal information; or enable attackers to change the product, insert agents, or corrupt information. [19] Development controls to mitigate the potential for security flaws or vulnerabilities include security focused design reviews, multi-party implementation methods, and security testing at every phase of the development.

The common current practice of accepting software that satisfies functionality requirements without regard for security properties is obviously not sufficient. Software assurance must be a primary concern of all acquisition managers, be they program or project managers, buyers, integrators or specifiers of systems requirements. The fact that prevailing practices of software suppliers have failed to produce the safe, secure, reliable, and dependable software can no longer be tolerated. As Judge Learned Hand said in the famous case The T. J. Hooper, "Indeed in most cases reasonable prudence is in fact common prudence; but strictly it is never its measure; a whole calling may have unduly lagged in the adoption of new and available devices. It never may set its own tests, however persuasive be its usages. . . .[T]here are precautions so imperative that even their universal disregard will not excuse their omission." [20]

Assuming that a decision to contract was made, the acquisition manager uses the outputs of the Initiation and Planning process as inputs into the Contracting process. . Of course, software assurance may be approached differently based on the type of acquisition. Differences, where applicable, must be addressed for custom software development, commercial-of-the-shelf software (COTS), system integration services, and IT services and business process outsourcing. [21] The Contracting phase includes the Requests for Proposals (RFPs), the responses to the RFPs (both proposals and requests for clarifications), and finally source selection and contract negotiation and finalization.

RFPs contain the Work Statement, Terms and Conditions, and Instructions to Suppliers. The Work Statement should include the initial software and software assurance requirements. These requirements can include a Software Assurance Plan, Software Risk Management Plan, and an Assurance Case Management Plan. Terms and Conditions should be appropriate for the type of software. This can include appropriate software assurance acceptance criteria. It is important that Instructions to Suppliers are clear and unambiguous regarding what they are to submit in their proposals. The acquisition manager should not only ask for documentation that provides evidence of potential suppliers' ability to delivery on software assurance requirements but also provides evidence concerning foreign ownership, control or influence.

Responses to the RFPs must be examined to ensure that the software assurance requirements are understood and not minimized. Suppliers should take care in addressing all the software assurance requirements and request for evidence.

Source Selection and Contract Negotiation and Finalization are the closing process in this phase. The acquisition manager and team of experts, including software assurance, evaluate the evidence that is submitted by the supplier in order to identify the supplier that can best provide the services. It is vitally important that the contract negotiation include consideration of security requirements performance issues on par with other negotiated elements.

Following the Contracting phase, the project moves to Implementation.

VIII. Acquisition—Implementation

This phase is a critical phase in software assurance. Acquisition managers must ensure that all the software

assurance requirements are adequately implemented. The Implementation phase includes:

- Project/Contract Management. Project and contract managers must rely on software assurance experts to ensure that the software assurance requirements are implemented appropriately.
- Software Assurance Plan. Software assurance experts ensure that the implementation is in accordance with the Software Assurance Plan.
- Software Risk Management. Software risks are monitored in accordance with the Software Risk Management Plan.
- Assurance Case Management. A Software Assurance Case is a reasoned, auditable argument created to support the contention that the defined software will satisfy software security requirements and objectives. [22]
- Software Assurance Acceptance Criteria. Acceptance criteria may be included in a term or condition or as part of the assurance case.

Following the Implementation phase, the project moves to Sustainment.

IX. Acquisition—Sustainment

Sustainment is the logistics tail in the acquisition of software. Most software endures years of sustainment. Therefore, it is important not to compromise software assurance during this phase and appropriately manage the change in security requirements. The Sustainment phase includes:

- Change Control. Change must be managed to ensure that the software security functions are maintained at the appropriate level of acceptable risk.
- Software Risk Management. Implementation does not end the need to continue to manage software risk. The Software Risk Management Plan used to manage risk during the Contracting phase should continue to be used and updated.
- Assurance Case Management. Just as software risk management continues during Sustainment, the software assurance case must be maintained and updated.

X. CONCLUSIONS

Our increasing dependency on cyberspace and the systems and networks that comprise its operational environment makes us increasingly vulnerable to failures of this critical infrastructure, and to cascading failures that could affect other critical infrastructures that are interdependent with the information infrastructure. Thus, assuring the security and reliability of software has become critical to our national and economic security. Efforts now underway are attempting to define methodologies for defining, developing and acquiring, and sustaining software for which there is a level of confidence that the software functions as intended and is free of vulnerabilities, either intentionally or unintentionally designed, as part of the software or inserted later. The *Software Assurance Guide for Acquisition Managers* is one such effort that will lead to trusted software that does what it is designed to do, and does not do anything other than what it was designed to do.

XI. References

N. B. References to Polydys, M.L. and Wisseman, S.. (2006) *Software Assurance Guide for Acquisition Managers, Version 0.1*. Washington, DC: Software Assurance Acquisition Working Group, are to a work in progress. Page numbers may change as the document is prepared for publication.

[1] Bush, George W. (2003) *National Strategy to Secure Cyberspace*. Washington, DC: The White House.

[2] UU.S. President's Information Technology Advisory Committee. (February 2005). *Cyber Security: A Crisis of Prioritization.* National Coordination Office for Information Technology Research and Development: Arlington, VA, p. 9.

[3] Dyson, Freeman (15 May 1998) *Science*: Vol. 280. No. 5366, pp. 1014 – 1015, accessed March 3, 2006, on-line at http://www.sciencemag.org/cgi/content/full/280/5366/1014?view=full .

[4] *National Strategy to Secure Cyberspace,* p. viii.

[5] IEEE Standard 610.12.

[6] *National Strategy to Secure Cyberspace,* p. 35.

[8] Graff, Mark & van Wyk, Kenneth (2003) *Secure Coding: Principles and Practices*. Sebastopol, CA: O'Reilly & Associates.

[9] Forno, Richard (December 2001) "Who Needs Hackers? We've got Microsoft" accessed March 3, 2006 on-line at http://www.infowarrior.org/articles/2001-15.html.

[10] Polydys, M.L. and Wisseman, S.. (2006) *Software Assurance Guide for Acquisition Managers, Version 0.1*. Washington, DC: Software Assurance Acquisition Working Group. p. 11.

[11] GAO-04-678 (May 2004) Defense Acquisitions: Knowledge of Software Suppliers Needed to Manage Risks. Washington, DC: General Accounting Office.

[12] *Ibid*, p. 19.

[13] See Graff & van Wyk, supra note 7. Also, Howard, Michael & LeBlanc, David (2004) *Writing Secure Code*, 2nd Ed. Redmond, WA: Microsoft Press.

[14] Polydys, p. viii.

[15] Polydys, p.

[16] Federal Information Processing Standard (FIPS) Publication 199 (February 2004). *Standards for Security Categorization of Federal Information and Information Systems.* Gaithersburg, MD.: National Institute of Standards and Technology (NIST), U.S. Department of Commerce.

[17] Redwine, S. T., Baldwin, R. O., Polydys, M. L., Shoemaker, D. P., Ingalsbe, J.A., Wagoner, L.D. (2006). Secure *Software Assurance: A Guide to the Common Body of Knowledge to Produce, Acquire, and Sustain Secure Software.* Department of Homeland Security: Washington, DC.

[18] Polydys, p. 15.

[19] Polydys, p. 12.

[20] The T. J. Hooper. The Northern No. 30 and No. 17. The Montrose. in re Eastern Transp. Co. New England Coal & Coke Co. V. Northern Barge Corporation. H. N. Hartwell & Son, Inc., v. SAME.
60 F.2nd 737 (U. S. Ct. App., 2nd Circuit, 1932)
cert. denied 287 U.S. 662; 53 S. Ct. 220; 77 L. Ed. 571; 1932 U.S. LEXIS 387 (1932).

[21] No Author (December 2005). *The Impact of Software Assurance on the Procurement Process*. INPUT, TargetVIEW, Volume 1, Issue 10. INPUT: Reston, VA.

[22] Redwine, p.111.