Carnegie Mellon University
Software Engineering Institute

Program Manager's Guidebook for Software Assurance

Dr. Kenneth E. Nidiffer
Dr. Carol Woody
Timothy A. Chick

December 2018

SPECIAL REPORT
CMU/SEI-2018-SR-025

Software Solutions and CERT Divisions

http://www.sei.cmu.edu

REV-04.06.2018.0

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

This report was prepared for the SEI Administrative Agent AFLCMC/AZS 5 Eglin Street Hanscom AFB, MA 01731-2100

DM18-1004

Table of Contents

List of Figures

List of Tables

Acknowledgments

The development of this guidebook was sponsored by the Department of Defense Joint Federated Assurance Center (JFAC). The mission of the JFAC is to promote software and hardware assurance in defense acquisition programs, systems, and supporting activities.

We thank Thomas Hurt from the Office of the Deputy Assistant Secretary of Defense (Systems Engineering) (DASD(SE)) and Mike McLendon from the Carnegie Mellon University Software Engineering Institute (SEI), as well as many other individuals from different organizations who contributed to the development of this guide. Specifically, we thank Roy E. Wilson, Professor of Acquisition Cybersecurity at the Defense Acquisition University, for his leadership in the development of the Defense Acquisition University CLE 081 Software Assurance Course, which we closely followed in the development of this guidebook.

We also thank the Guidebook Steering Committee: Dr. William Nichols and Dr. Thomas Scanlon from the SEI; Dr. Scott M. Brown and Bradley Lanford from Engility Corporation in support of DASD(SE); Robert (Bob) Martin from The MITRE Corporation in support of DASD(SE); Dr. David A. Wheeler from the Institute for Defense Analysis in support of DASD(SE); Carol Lee from the Navy; and Michele Moss from Booze Allen Hamilton in support of the Office of the Secretary of Defense, Chief Information Officer.

We acknowledge David Wheeler and Bradley Lanford for their help in providing many of the DASD(SE) diagrams reprinted with permission in this guidebook, as well as providing the basis from which others were derived.

We also thank Scott Brown and Thomas Hurt as the authors of the initial Project Manager Checklists, in association with DASD(SE), which provided the basis for the checklists found in Appendices D and E.

Finally, we thank the JFAC Software Assurance Technical Working Group members for their review, comments, and suggestions.

Abstract

The *Program Manager's Guidebook for Software Assurance* supports project managers who must integrate software assurance engineering activities into the acquisition lifecycle. The goal of the guidebook is to help the program manager (PM) understand and address the software assurance responsibilities critical in defending software-intensive systems. It presents actions a PM must take to ensure that software assurance is effectively addressed. These actions require an understanding of program mission threads, threat awareness, and the roles and responsibilities of members of the program office team. The guidebook objectives are aligned with (1) Enclosure 14 of Department of Defense (DoD) Instruction 5000.02, which provides policies and principles for cybersecurity in defense acquisition systems; (2) the Defense Acquisition University's Software Assurance Course (CLE 081); (3) the DoD Integrated Defense Acquisition, Technology, and Logistics Lifecycle; and (4) the Deputy Assistant Secretary of Defense (Systems Engineering) Software Assurance Concept of Operations.

1 Introduction

Who should use this guidebook, and why?

This guidebook should be read by program managers (PMs) of software-enabled defense systems, especially weapon systems, and their stakeholders, including members of their program teams who are assigned responsibilities related to software assurance. It focuses on supporting the program management team in addressing engineering activities that reduce exposure to software vulnerabilities, in a timely manner and without dramatically affecting the cost, schedule, or performance of the defense system.

The goal of the guidebook is to help PMs understand and address the software assurance responsibilities critical in defending software-intensive systems. A range of software assurance policies, roles, responsibilities, practices, procedures, and concepts are available to support PMs in handling software assurance decisions throughout the acquisition lifecycle.

1.1 Defining Software Assurance in a Changing Landscape

This guidebook uses the following definition of *software assurance*, taken from the National Defense Authorization Act (NDAA) of 2013:

> *The term "software assurance" means the level of confidence that software functions as intended and is free of vulnerabilities, either intentionally or unintentionally designed or inserted as part of the software, throughout the lifecycle* [P.L. 112-239 2013, Sec. 933(2)].

This publication provides guidance according to current Department of Defense (DoD) policy and for a generalized acquisition environment applicable to most PMs. PMs are required by DoD policy to integrate software assurance into the system acquisition lifecycle. DoD Instruction (DoDI) 5000.02 requires that "Software assurance vulnerabilities and risk-based remediation strategies will be assessed, planned for, and included in the Program Protection Plan (PPP)" [DoD 2017, p. 91]. The PPP draws its definition for software assurance from Public Law 112-239. Current policies, standards, and regulations related to software assurance are provided in Appendix C.

This guide assumes that a PM

- understands that software assurance guidance may change if DoD acquisition policy changes
- has a workforce with core competencies in systems engineering and software engineering sufficient to implement system and software assurance activities to achieve the program's software assurance goals
- will tailor assurance activities to meet his or her program's specific needs

1.2 Using This Guide

The guidebook structure follows the structure of Enclosure 14: DoD Instruction, which provides the PM with cybersecurity guidance in a defense acquisition system by acquisition phase. For example, for a given phase, Enclosure 14 informs the PM about what to accomplish. It also provides additional materials in support of this guidance. Each phase has its own set of processes for primary focus. The PM must assess the importance and cost of process activities in each phase (which may be replicated across phases and may occur more than once in a single phase) relative to the achievement of the program's software assurance goals. For example, system engineering and software development processes will nominally occur in each phase and, depending on the phase, might be repeated more than once within a single phase.

The quick-start guide in Section 2 summarizes the PM's responsibilities for software assurance throughout the development and sustainment lifecycle, and Section 3 explains how software assurance relates to other program security responsibilities. Section 4 identifies ways that PMs can find the software assurance requirements for their programs and why they need to focus on software assurance throughout the process. Section 5 lists roles that program team members and others might play and explores program management office (PMO) structures that can be used to address software assurance. Section 6 identifies the objectives that PMs should consider at each phase in the acquisition lifecycle, while Section 7 focuses on software assurance progress at major milestones. Finally, Section 8 explains how PMs can plan and control program activities to achieve the program's software assurance objective.

The guidebook also contains supplemental information in the appendices, including further description of applicable policies and documents, checklists, tool guidance, and references.

2 Quick-Start Guide

What responsibilities do program managers have for software assurance?

Large portions of system functionality are shifting from hardware to software to capitalize on the increased flexibility and speed of component delivery. However, with these benefits come other challenges that must be addressed.

The PM is accountable for ensuring that software vulnerabilities are addressed. The PM can expect cost and schedule impacts if vulnerabilities are not identified until verification and validation activities are performed. Vulnerabilities should be identified and addressed as software is designed and developed.

2.1 Address Assurance Early to Avoid Exponential Cost Growth

Software contains vulnerabilities, many of which may be accessible to exploit. Approximately 84% of recorded software breaches exploit vulnerabilities at the application layer [Clark 2015]. As software exerts more control over functionality in complex systems, the potential risk posed by software vulnerabilities increases in kind. The later in the lifecycle that vulnerabilities are discovered, the costlier it is to correct them. Finding vulnerabilities early is especially important because engineering-related vulnerabilities are not easily corrected after a system has been deployed. If software-reliant systems with known software vulnerabilities are allowed to operate, they pose a high degree of residual cybersecurity risk, putting the associated operational missions in jeopardy.

System verification and validation steps are expanding to include looking for these types of vulnerabilities, but the ability to find them in test is limited. Conventional (mostly positive) test is typically only 40–50% effective [Jones 2011], and security-related vulnerabilities can be even more difficult to identify. More distressingly, 15% of defect fixes introduce new defects, of which only about half will be detected. The fact that defects often mask other defects by altering execution behavior compounds the problem. Finding substantial numbers of defects and vulnerabilities during test should be a red flag indicating that additional vulnerabilities remain undetected.

2.2 Address Software Assurance Throughout the Lifecycle

To truly protect DoD systems, engineering must ensure that software assurance is addressed throughout development and sustainment activities. Because 3–5% of all defects are potential security vulnerabilities [Woody 2014], a focus on finding and removing vulnerabilities throughout the acquisition lifecycle, just as one does with quality defects, is the only cost-effective approach to addressing software vulnerability issues.

For example, using static analysis tools to evaluate code or binaries prior to integration reduces overall project costs [Nichols 2018]. However, costs increase when using static analysis tools only after integration. As a result, DoD policy requires PMs to integrate software assurance into the system acquisition lifecycle.

System engineering's role is to ensure that the system meets the assigned requirements, which includes ensuring that the system functions as intended. Security is a quality attribute that must be balanced with other system needs such as safety, usability, and maintainability. As increasing amounts of system functionality are designated to software, the impact of software vulnerabilities also increases. By integrating software assurance throughout the lifecycle, the PM will have sufficient evidence that vulnerabilities have been identified and addressed.

2.3 Collect Evidence to Support Assurance Claims

The NDAA for fiscal year 2013 defines software assurance as the "level of confidence that software functions as intended and is free of vulnerabilities, either intentionally or unintentionally designed or inserted as part of the software, throughout the lifecycle" [P.L. 112-239 2013, Sec. 933]. Establishing this level of confidence requires collecting and analyzing evidence across the lifecycle to confirm that the software "functions as intended" and "is free from vulnerabilities." From design to implementation, many participants in the lifecycle touch the software in some form. Evidence needed to establish confidence is widely distributed across all of these sources, including contractors.

The PM must ensure that an appropriate range of evidence is collected to demonstrate both the functionality of the software and the absence of vulnerabilities. Since both of these activities typically require the assignment of funding and time on the schedule, the PM must ensure that the necessary resources, tools, and processes are incorporated into the selected lifecycle and program plan to address both parts of software assurance.

Many software decisions are not within the PM's direct control. At times the selection and development of software are several layers into the acquisition supply chain. The PM must ensure that solicitations and contracted requirements include the requirement to address the criticality of software assurance for the system. Likewise, the PM must ensure that acceptance criteria include mechanisms to confirm that software assurance has been addressed by those actually building or purchasing the software. These criteria include retaining the government's unlimited data/software rights and validating that all required data needed to rebuild any executable software was delivered [SMC 2018].

Suppliers can provide a wide range of evidence about the software they incorporate into their delivered products, including that of their subcontractors. However, requests may have cost and schedule impacts. The PM must ensure that suppliers plan for providing sufficient evidence to establish confidence that they have addressed software assurance.

The evidence needed by the PM to establish confidence for software assurance may not always require a specific allocation of effort and cost. Using the Risk Management Framework (RMF) to meet requirements for quality, safety, anti-tamper (AT), and information assurance (IA), a program can benefit from pooling software assurance activities with activities to meet other assurance requirements. Such efficiencies in testing and information gathering can reduce cost and schedule impacts through coordinated planning and information sharing. The evidence collected

to show that software vulnerabilities have been addressed can also support risk management steps needed to obtain the authority to operate (ATO).

2.4 Designate Roles for Software Assurance Responsibilities

Effective software assurance requires that the PM designate an engineer to take responsibility for managing the software assurance in a system throughout the lifecycle. This engineer will have planning, monitoring, and verification responsibilities to support the PM's responsibilities for software assurance. The person in this role must

- be familiar with the planned and selected software in use in the system
- have access to the range of participants, including contractors, touching the software and the authority to request and collect evidence
- be familiar with the system lifecycle and understand what evidence can be collected and when
- understand the potential risks that reduced levels of confidence could trigger

Depending on the lifecycle choices made in acquisition, there are many potential candidates for this role. (See Section 5 for more information about roles and responsibilities.)

2.5 Create and Manage Plans, Reports, and Artifacts

Program Protection Plan and System Engineering Plan

DoD 5000.02 requires that "Software assurance vulnerabilities and risk-based remediation strategies will be assessed, planned for, and included in the Program Protection Plan (PPP)" [DoD 2017]. The PM must also report the plan for collecting software assurance evidence as part of the System Engineering Plan (SEP). The plan includes how the system will be engineered to address requirements and, as part of the PPP, how software vulnerabilities will be identified and addressed. The PM is required to incorporate technical approaches from contractors for addressing software assurance as they are approved.

Required Software Components

The delivered system must include software code for all components, including open source software, commercial products, libraries, code developed by third parties, and any other integrated code. Instructions for assembling the components into the delivered operational software are also needed. Without all of these pieces, the ability to repair and rebuild the operational software to address future changes and fix missed vulnerabilities will be uncertain.

The PM must ensure that the delivered system, including the software code for all components, can be independently verified and validated to have sufficient software assurance. Independent verification and validation (IV&V) must be able to assemble the operational software from the delivered code components and apply tools to confirm that the code is free from vulnerabilities. This confirmation process can only be achieved if the PM appropriately addresses the government's rights to technical data and computer software in the contract [SMC 2018] and enforces the delivery of all data, including all software components, associated with enforcement of the government's rights. Without the source code and all associated components needed to reproduce all delivered executables, the ability to identify and address vulnerabilities is extremely limited. In

addition to enabling software assurance activities, insisting on unlimited government-purpose rights can reduce total ownership costs, a goal that is recognized and endorsed by law. In accordance with NDAA 2018, Section 871 [P.L. 115-91 2017],

> *As part of any negotiation for the acquisition of noncommercial computer software, the Secretary of Defense shall ensure that such negotiations consider, to the maximum extent practicable, acquisition, at the appropriate time in the life cycle of the noncommercial computer software, of all software and related materials necessary—*
>
> 1. *to reproduce, build, or recompile the software from original source code and required libraries;*
> 2. *to conduct required computer software testing; and*
> 3. *to deploy working computer software system binary files on relevant system hardware.*

2.6 Consider Risks in All Sources of Software

Due to its high costs, software is rarely built to meet specific requirements but is selected instead from a wide range of available commercial products, language libraries, code-generation tools, legacy systems with a similar purpose, and other types of reusable software. The selected elements are then assembled to perform the desired functionality. With any of these assembled options, decisions about software assurance were made for other circumstances that typically do not match the needs of the current system.

Opportunities for unneeded and undesirable functionality and vulnerabilities abound, which create risks for the assembled system. The PM must ensure that assessments are performed on code and binaries from all sources to identify and remove vulnerabilities in a timely manner. Since software is continually updated, lifecycle processes and practices must be in place for updates from all of these many sources in order to address future software vulnerabilities.

2.7 Budget for and Select Appropriate Tools

The PM must understand the challenges in identifying software vulnerabilities as they impact cost, schedule, and the resulting software assurance for a system. Throughout the lifecycle, many participants make choices about the tools used to find vulnerabilities, when in the lifecycle they are executed and by whom, the skill levels of the tool users, and their effectiveness in addressing the vulnerabilities identified by the tools.

The PM ensures that there is sufficient investment to identify and address vulnerabilities that represent a high risk to mission success. Available tools for evaluating the assurance of existing software are improving, but no one tool is sufficient, and each tool is only effective for a select number of vulnerabilities. Tools also produce many false vulnerability identifications, and research effort is required to determine which are of actual concern. Tools include a generalized structure for prioritizing vulnerabilities, but that prioritization may not match the risk to a system that also includes factors such as accessibility and potential mission impact. The point in the lifecycle where tools are applied and vulnerabilities are identified will also impact cost and schedule; the repair needed for a vulnerability will require less effort earlier in the lifecycle when

changes do not require extensive retesting. Appendix H provides a table of tools mapped to the lifecycle stages where they are considered useful. This gives the PM a sense of the range of potential tools that can be used. Because of the volume of software and the complexity of identifying vulnerabilities, simple human review is insufficient and tools should be used where possible to improve results at scale.

2.8 Determine Effectiveness of Software Assurance Activities

The PM will know that software assurance is effectively addressed when the following are observed:

1. Processes and practices consistently provide evidence that software vulnerabilities are identified early, using a range of tools for all types of software incorporated into a system.
2. Vulnerabilities are identified and addressed.
3. Later lifecycle steps show reduced levels of software vulnerabilities.

2.9 Summary

The PM is accountable for software assurance. In short, the PM must thoroughly understand and plan for the following responsibilities:

1. Determine who in engineering is responsible for software assurance, and make sure they have the training, experience, and authority to address the range of software included in the system.
2. Make sure all software has been identified and plans have been made for evaluating it to identify software vulnerabilities using appropriate tools. This includes all acquired software products, developed code, language libraries, code included in acquired system components, and firmware.
3. Plan for collecting the evidence needed to confirm that software assurance has been addressed, and understand how the evidence will be requested, verified, and retained.
4. Know how to report and document software assurance evidence to establish confidence that the software will function as intended and software vulnerabilities have been identified and addressed.
5. Ensure that the government maintains and enforces unlimited government-purpose rights. These rights are essential in enabling the PM to fix vulnerabilities and reduce security risks to the program throughout its life. The trading of data rights against cost and risks is unacceptable. Without unlimited government-purpose rights associated with the computer software, the total cost of ownership is unbounded, resulting in the acceptance of unlimited costs and risks.

3 Software Assurance Concepts

How does software assurance relate to other program security responsibilities?

PMs face intense software assurance challenges from complex and changing requirements, technology, and agency and stakeholder dynamics. The PM's objective is mission success; thus, the achievement of software assurance is an important goal.

3.1 Security Activities Within a Program Management Office

Security concerns are focused on controls established to protect a system from compromise. Software assurance supports security through the removal of the following:

- design and coding weaknesses that hinder a system from functioning as intended
- vulnerabilities that can allow controls to be bypassed

Activities that focus on these critical concerns support both security and software assurance.

There are two broad areas of concentration from a technical perspective:

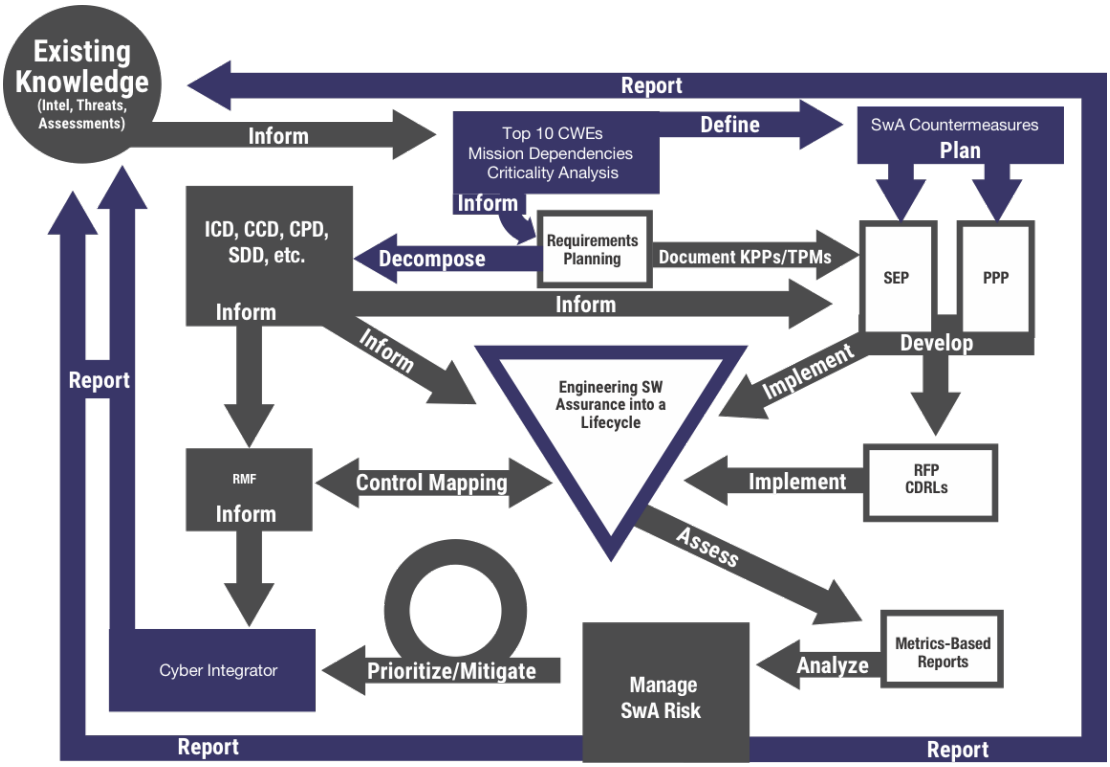
1. development process – assurance countermeasures conducted during the development process to mitigate and minimize attacks (e.g., threat assessment and modeling, attack surface analysis, architecture and design reviews, application of static and dynamic code assessment tools and services, penetration testing, and red teaming) that the developed system is likely to face when deployed into operation
2. operational system – attack countermeasures and other assurance activities applied within the operational environment (e.g., failover, fault isolation, encryption, application firewalls, least privilege, and secure exception handling) to mitigate attacks against the delivered system and software interfaces, which may include commercial off-the-shelf (COTS), government-off-the-shelf (GOTS), and other off-the-shelf software as well as open source software

To more fully cover both technical and security aspects of assurance, a companion guidebook has been published: *The DoD's Developer's Guidebook for Software Assurance*. The companion guidebook provides a broad focus because security must be maintained throughout all phases of the product and development lifecycle. While developers are mostly concerned with development and maintenance, they require a basic awareness of all processes for several reasons. Requirements and architecture place many constraints on development. Many products include COTS, GOTS, or open-source software components, so developers must be aware of risks introduced through acquisition and the supply chain. Transition increasingly includes monitoring the product in use to identify threats. Threats need to be mitigated throughout a system's life, which requires the ability to update and change the software after the initial deployment.

3.2 Software Assurance Concept of Operations

An endeavor as complex as developing or acquiring a software-enabled defense system has many components that interact with each other. Figure 1 shows an overview of the concept of operations for a DoD system. The central focus is the software assurance activities and work products

accomplished by the PM and other members of the program office in the acquisition lifecycle. In addition, there are reporting mechanisms and supporting analysis and input data that improve the PM's ability to reduce impacts to software assurance.



Derived with permission from a diagram originally developed by DASD(SE)

Figure 1: DASD(SE) Software Assurance Concept of Operations

The basis of understanding the expected weaknesses in a software system comes from the existing knowledge base of threats, vulnerabilities, and other existing intelligence information. Known vulnerabilities are documented in the National Vulnerability Database (NVD).[1] Threat modeling approaches [Shevchenko 2018] assemble system elements and the ways in which they can be compromised by leveraging weaknesses and known attack patterns. Based on the weaknesses and components specific to a system, countermeasures are selected to enable the mitigation of weaknesses in the software and all supporting software that may serve as an access point for the exploitation of software within a system. The plan for implementing countermeasures, key performance parameters (KPPs), technical performance measures (TPMs), and all supporting planning artifacts should be documented in the SEP and the PPP. (See Appendix B for more information about these plans.)

If the program is supported by a contractor, all software assurance activities should be identified in the Request for Proposal (RFP) and reportable in contract deliverables. Program acquisition

[1] See https://nvd.nist.gov/.

planning and contract documents should be used to guide implementation. As engineering decisions are made, systems engineers should be conscious of the RMF and ensure that engineering implementations are mapped to satisfied controls for review by cybersecurity professionals.

Metrics-based reporting should be used to track a program's progress against defined requirements and objectives. If a requirement is to analyze some percentage of software components for vulnerabilities, then selected metrics can include percentage of code scanned for specific vulnerabilities, percentage of reported vulnerabilities analyzed and remediated, percentage of software components evaluated for vulnerabilities, and so forth. In addition, these reports should be analyzed to inform the risk-based scoring of weaknesses and vulnerabilities. Programs should identify high-priority vulnerabilities that represent risks to software assets and monitor them closely, ensuring that selected tools will look for them and mitigation activities will address them. Reports should be captured in the PPP.

Although it is a large component of the system security engineering (SSE) risk for a program, software assurance is only one element of a program's overall SSE risk. At any stage of the mitigation process, a program can use the Cyber Integrator (see Section 5.4.1) to identify residual risk that remains unmitigated by ongoing SSE activities. This approach allows programs to effectively manage SSE risk and track progress of the mitigation of these risks.

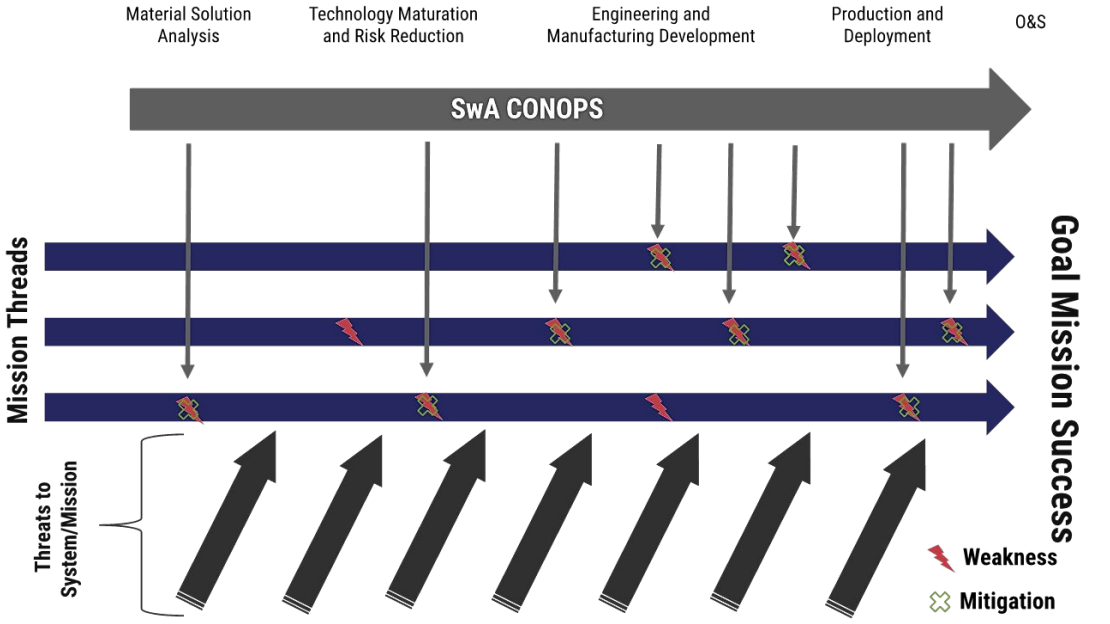
4 Mission Requirements and Threats

Where will the PM find the software assurance requirements for a program, and why is it important to know what they are?

The expanded use of software to address functionality in today's systems leaves them vulnerable to the impact of weaknesses that can be introduced throughout the acquisition lifecycle. Exploitable weaknesses in software can impact the successful execution of one or more mission functions assigned to a system. PMs need their systems and any interfacing systems to work successfully to achieve mission success.

A *threat* is a circumstance or event with the potential to adversely impact mission success. PMs may not understand the specific threats to their systems until mission threads are developed that show the system functions needed to perform a mission. Threat modeling, a process for determining the ways in which a threat can impact mission success, can be applied to the mission threads to identify system requirements that appropriately address mission impacts [Shevchenko 2018]. Many threats have been realized by exploiting software vulnerabilities.

Threat modeling methods can be used to reveal what computer software configuration items and computer software units are critical and require extra software assurance attendance. Figure 2 provides a view of software assurance from a mission-success perspective. In the figure, not all weaknesses (or threats) are mitigated. This is because not all weaknesses can be exploited or impact the mission, thus no resources are applied to mitigate them.



Derived with permission from a diagram originally developed by DASD(SE)

Figure 2: Conceptual View: Software Assurance Mission Success

4.1 Threat Sources

A system can enable threats in any number of ways. Some examples are described below.

- Threats exist throughout the supply chain, such as malware introduced into software at the point of development, or code that is intercepted and modified while being delivered to operational locations.
- Negligent users can enable threats, including those willing to click on a suspicious link in an email or who pick up a free USB drive from an exhibitor at a conference and plug it into their work computer. Another example could be maintenance staff who plug a phone into the USB port on a work computer.
- A system is loaded with wireless access points, not only the Wi-Fi or Bluetooth that people normally consider, but also satellite connectivity for beyond-line-of-sight communications and numerous radio frequency apertures.
- Removable media, including CD/DVD drives in the information system and removable mission/maintenance drives in the weapon system, can create security issues.

An adversary can exploit a weakness in a software system. A *weakness* is a shortcoming or imperfection in software code, design, architecture, or deployment that, under proper conditions, could become a vulnerability, or contribute to the introduction of vulnerabilities in DoD systems' software. The Common Weakness Enumeration (CWE) is a list of software weaknesses.[2] A *software vulnerability* is a collection of one or more weaknesses that contain the right conditions to permit unauthorized parties to force the software to perform unintended behavior (i.e., it is exploitable).

[2] See https://cwe.mitre.org/.

The Common Vulnerabilities and Exposures (CVE) resource is a publicly available and free list of standardized identifiers for common computer vulnerabilities and exposures.[3]

Examples of software weaknesses include

- buffer overflows and format strings
- structure and validity problems
- common special element manipulations
- channel and path errors
- handler errors
- user interface errors
- pathname traversal and equivalence errors
- authentication errors
- resource management errors
- insufficient verification of data
- code evaluation and injection
- randomness and predictability

4.2 Threat Impact

Threats can impact mission success by compromising mission data confidentiality, integrity, and/or availability (CIA). Mission success can also be impacted through compromised capabilities, such as authentication and non-repudiation. The following examples, adapted from the *Cybersecurity Test and Evaluation Guidebook* [DoD 2018a], illustrate a sample of software-related attacks that could impact the mission of a system in each of those categories.

- Confidentiality: An advanced persistent threat (APT) resides in the government or contractor's development environment. The APT exfiltrates lower level design information from unclassified computing systems or networks and Critical Program Information (CPI) or other information from classified networks. The adversary uses this information to copy the system or to develop countermeasures.
- Integrity: Malicious software or malware introduced into the weapon system IT or its embedded supervisory control and data acquisition (SCADA) system anywhere in its lifecycle could impact the integrity of the software or data, resulting in either a mission kill or a cyber-physical kill. Poorly designed or constructed code that introduces software weaknesses, such as buffer overflows, can also reduce integrity.
- Availability: An adversary develops denial-of-service attacks that could kill an information subsystem's ability to communicate with the DoD Information Network (DoDIN), which would result in a mission kill.

[3] See https://cve.mitre.org/.

- Authentication: A security measure is designed to establish the validity of a transmission, message, or originator, or a means of verifying an individual's authorization to receive specific categories of information. Lack of authentication may result in fraudulent interactions with unsuspecting systems, which can be used to exploit system vulnerabilities.
- Non-repudiation: The sender of data is provided with proof of delivery and the recipient is provided with proof of the sender's identity, so neither can later deny having processed the data.

5 Roles, Responsibilities, and Organizational Structures

What roles do program team members and others play in accomplishing software assurance and security objectives, and what PMO structures can be used?

The PM's objective is mission success; thus, the achievement of software assurance is a key goal and must include SSE activities performed by the program team as they relate to software assurance.

This section identifies members of the program office team, describes their roles and responsibilities, and specifies some of the people they might work with and the processes they might use. PMs need to understand these workforce responsibilities so that they can make appropriate planning and tradeoff decisions. Many program and project roles will have shared duties. Each role can include different team members working toward its objectives, depending on program structure and staffing choices. The right PMO structure can contribute to the effectiveness of the collaboration of various roles and responsibilities.

5.1 Program Manager

PMs are responsible for the software assurance and cybersecurity of their programs, systems, and information [DoD 2017]. The primary responsibilities of the PM are described in Section 2 of this report. PMs should view their software assurance activities—including cybersecurity and IA activities—as ongoing program activities to achieve the level of confidence needed for mission success. This approach will include a variety of key stakeholders who will have various roles within the program, depending on its scope.

The PM is concerned with how to efficiently and effectively identify risks and mitigate them so that mission success is achieved within the program's constraints. An objective of software assurance is to ensure that the software-intensive systems produced are more secure and help identify and mitigate associated risks. This is a significant challenge given the perspectives of the different members of the program team, related acquisition policy and guidance, and interrelations among security issues.

The PM should address software assurance requirements as part of the system's overall requirements set. Traditional risk themes do not adequately address programmatic software assurance tradeoffs, due in part to the recent rise in importance of addressing vulnerabilities. For example, many software assurance requirements are cross-cutting (e.g., architectural properties) and can emerge if a system is connected to other systems. The PM, especially on programs that are in a cost- and schedule-constrained environment, may want to use a System Security Working Group (SSWG) to identify a prioritized list of software assurance requirements to consider.

The systems engineer and/or system security engineer is responsible for leading and facilitating cross-discipline teams to conduct the SSE analysis necessary for development of the PPP. The cross-discipline interactions reach beyond the SSE community to the test and logistics communi-

ties. The PM can ensure that appropriate interactions and considerations are made for software assurance and the program's overall PPP through the establishment of a SSWG, as described in Section 5.6.

PMs should be aware that security classification and other issues may inhibit the prompt identification of vulnerabilities in their systems. For example, if a system essential to national security is vulnerable (e.g., a design flaw in a deployed missile) and that vulnerability could be exploited by an adversary, then the information about the vulnerability will be classified. The methodology for handling vulnerabilities discovered in other systems is hindered by immediate classification after discovery. The problem is that the findings are often immediately transferred to a higher level than the PM's system. It is therefore important that PMs work with their Intelligence Liaison Office and the Vulnerabilities Equities Policy and Process to ensure their program's vulnerabilities are adequately identified so they can be addressed.

5.2 Contracting Officer

The responsibilities of a contracting officer are detailed in Federal Acquisition Regulation (FAR) (48 CFR), Part 1.602-2:

> *Contracting officers are responsible for ensuring performance of all necessary actions for effective contracting, ensuring compliance with the terms of the contract, and safeguarding the interests of the United States in its contractual relationships.*

Contracts must include clauses specifically requiring that systems and software to be delivered are adequately secure for their purpose, including the appropriate use of software assurance approaches to address vulnerabilities. If this requirement is not incorporated into the contract, then the acquirer is at risk of not receiving secure systems and software. Thus, it is vital that contracts include such clauses, and it is vital that contracting officers ensure that the contracts they let include clauses that require systems and software to be delivered as adequately assured for their missions.

In terms of software assurance, the contracting office must ensure that the government maintains unlimited government-purpose rights to all data and that all computer software and associated computer databases and documentation are delivered. In accordance with Defense Federal Acquisition Regulation Supplement (DFARS) 252.227-7014, *Rights in Noncommercial Computer Software and Noncommercial Computer Software Documentation*, the term *computer software* means computer programs, source code, source code listings, object code listings, design details, algorithms, processes, flow charts, formulae, and related material that would enable the software to be reproduced, recreated, or recompiled. Computer software does not include computer databases or computer software documentation.

Based on this definition, the PMO should be able to independently reproduce, recreate, or recompile the delivered source code to independently validate that the contractor has met the contract deliverable requirement. Without the computer software, the PMO is unable to fix vulnerabilities and reduce security risks to the program throughout the program's life. The trading of unlimited government-purpose rights against cost and risks is unacceptable. Without unlimited rights, the total cost of ownership is unbounded, resulting in the acceptance of unlimited costs and risks. It is

strongly recommended that the DFARS 252.227 clauses listed in Table 1 be included in all solicitations and contracts [DISA 2018].

Table 1: Recommended DFARS Clauses/Provisions

Clauses/Provisions 252.227	7013	7014	7015	7016	7017	7019	7028	7030	7037
Solicitations	X	X	X	X	X	X	X	X	X
Contracts	X	X	X	X		X		X	X

The DASD(SE) sponsors an effort to support software assurance contracting requirements, which is organized through the DoD Software Assurance Community of Practice Contract Language Working Group. For the latest examples and recommendations of contracting language associated with software assurance, contact the DoD Joint Federated Assurance Center (JFAC; see Appendix A). Another resource is *Acquiring and Enforcing the Government's Rights in Technical Data and Computer Software Under Department of Defense Contracts: A Practical Handbook for Acquisition Professionals* [SMC 2018].

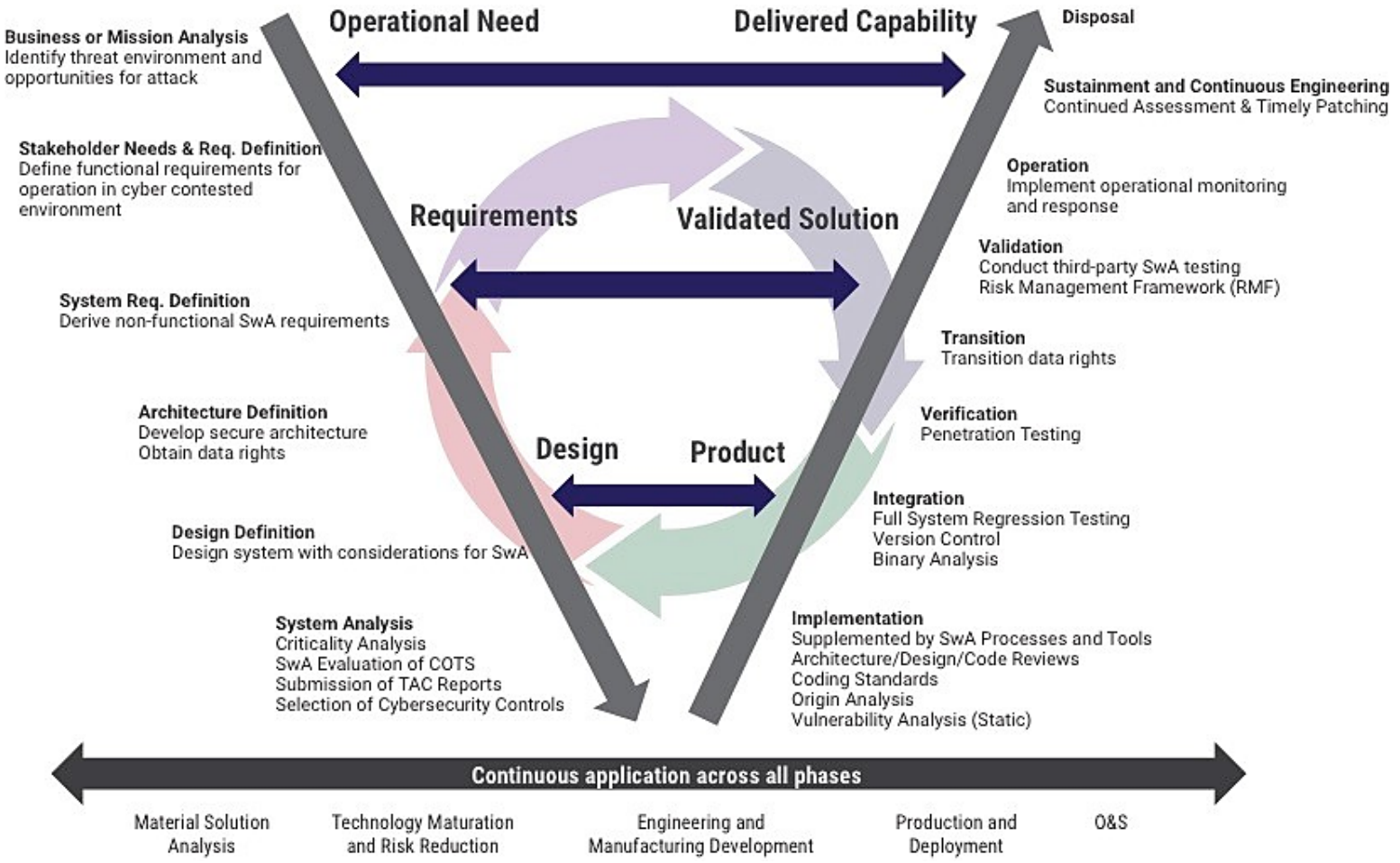
5.3 System Engineer

The system engineer, who may actually have other job titles, applies a methodical and disciplined approach to the specification, design, development, realization, technical management, operations, and retirement of a system. This role integrates and balances contributions from many engineering disciplines as the PM and lead systems engineer work with other members of the program office team to ensure that these activities are accomplished and appropriately resourced.

System engineering processes are tailored and applied as appropriate at every stage of the acquisition lifecycle, as presented in Figure 3. It is important to note that the technical and management processes of system engineering are not sequential. For example, these processes may be iterative, recursive, or concurrent. As a result, there are multiple "Vs" over the acquisition lifecycle.

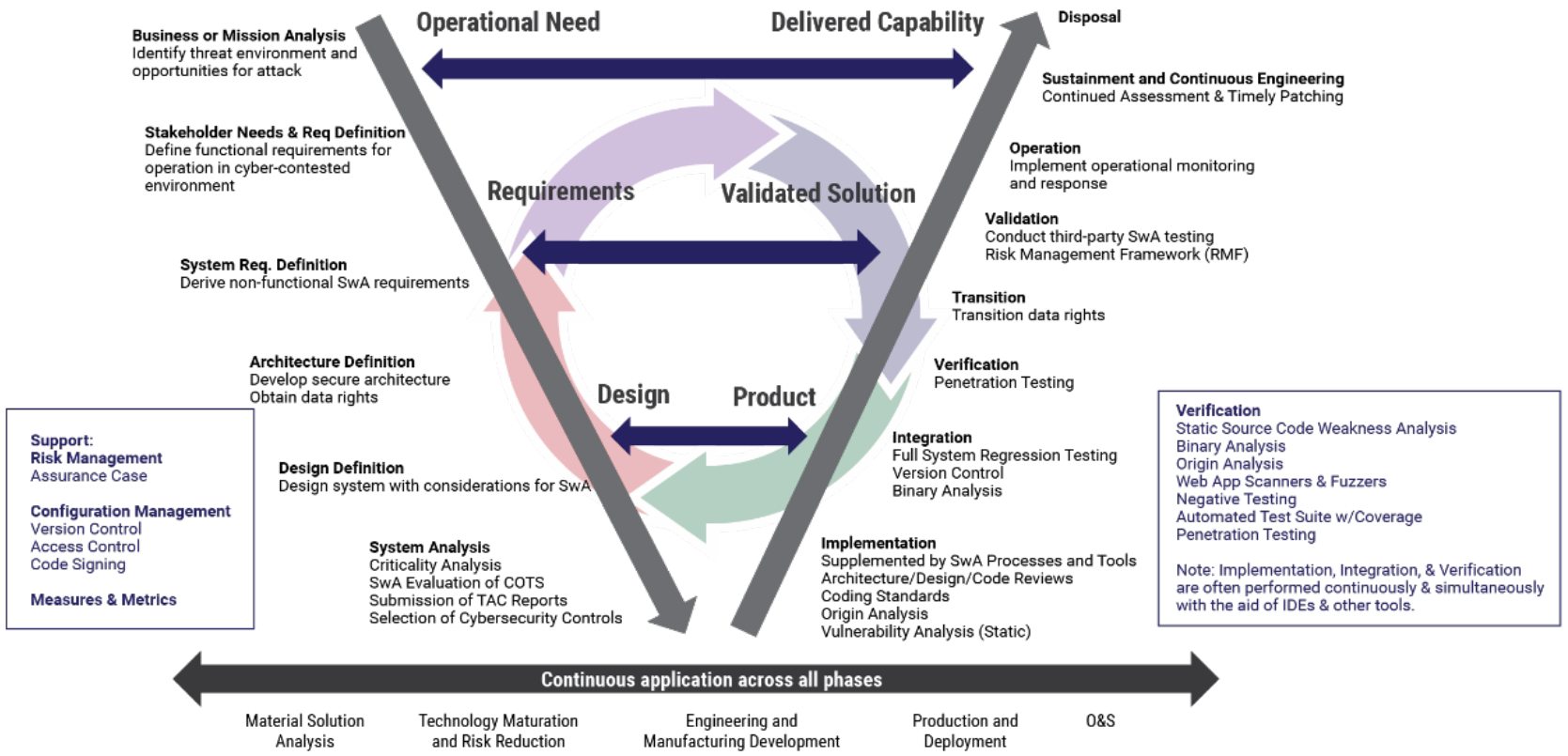
Representative software assurance activities can be mapped onto the system engineering V-model as presented in Figure 4. A uniqueness of software development compared to other types of development is that implementation, verification, and integration can occur in parallel and are present in all phases of the acquisition lifecycle.

An objective of software assurance is to ensure that the software-intensive systems are less vulnerable to attack. Software-intensive systems are increasingly complex and may contain design and coding flaws. The system engineer should be familiar with the use of tools that address preventive dynamic and static analyses to identify potential vulnerabilities. A holistic, system-level understanding of mission impact is recommended to appropriately recognize vulnerabilities that represent critical system operational risks and jeopardize operational mission success. Design flaws can account for a significant number of code vulnerability problems.



Derived with permission from a diagram originally developed by the Defense Acquisition University [Butler 2016]

Figure 3: System Engineering Model-V Diagram



Derived with permission from a diagram originally developed by DASD(SE)

Figure 4: Software Assurance Activities Mapped onto the System Engineering V-Model

5.4 Systems Security Engineer

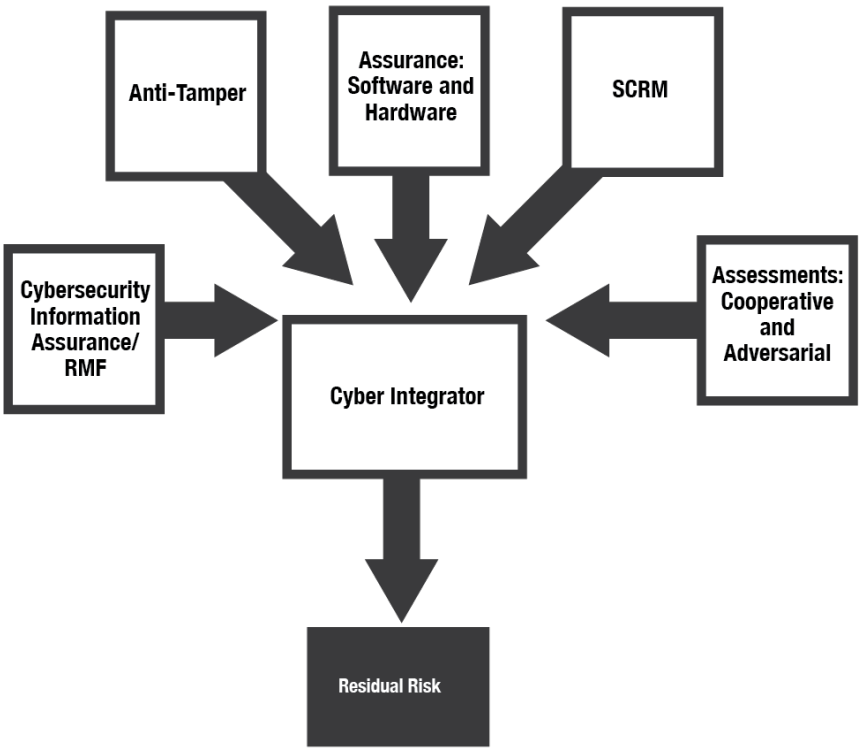
The systems security engineer, who may have several different job titles, is an interdisciplinary resource to enable the realization of secure systems. This role focuses on defining customer needs, security protection requirements, and required functionality early in the system development lifecycle; documenting requirements; and then proceeding with design, synthesis, and system validation while considering the complete problem. In accordance with DoDI 5200.44, SSE is an element of system engineering that applies scientific and engineering principles to identify security vulnerabilities and minimize or contain risks associated with these vulnerabilities [DoD 2012].

The systems security engineer supports program protection, which is the integrating process for managing risks to advanced technology and mission system functionality from foreign collection, design vulnerability, supply-chain exploit/insertion, and battlefield loss throughout the acquisition lifecycle. The effort of program protection is focused on Critical Program Information (CPI). CPI includes elements or components of a research, development, and acquisition program that, if compromised, could cause significant degradation in mission effectiveness; shorten the expected combat-effective life of the system; reduce technological advantage; significantly alter program direction; or enable an adversary to defeat, counter, copy, or reverse-engineer the technology or capability [DoD 2015b]. Since compromises can be aided by vulnerabilities, the role of the systems security engineer is closely aligned with the goals of software assurance.

5.4.1 Cyber Integrator

Effective software assurance and cybersecurity in DoD acquisition programs are top concerns for both DoD PMs and the DoD as a whole. An emerging concept to help DoD PMs meet these challenges is the establishment of a Cyber Integrator (CI) at the Program Executive Office (PEO)/Major Defense Acquisition Program (MDAP) level, to help address software assurance and cybersecurity risk in DoD acquisition programs. For many groups in the DoD, RMF is associated with cybersecurity, and engineering is associated with software assurance. The purpose of the CI is to coordinate software assurance and cybersecurity efforts within the PEO/MDAP, and that role includes effectively integrating processes and practices across all functional domains and acting as principal advisor to the PM on all software assurance and cybersecurity matters. A CI will not mitigate all the challenges faced by DoD PMs, but based on the emerging results of an ongoing Aviation and Missile Research, Development, and Engineering Center pilot program, the CI concept appears to be a step in the right direction.

Although it is a growing component of addressing the development and acquisition risk for a program, software assurance is only one element of a program's overall operational risk mitigation plan. Tradeoffs must be made when risk responses from different types of risk are in conflict. For example, safety may require a door to fail "open" and security may require it to fail "closed." At any stage of the mitigation process, a program can use the CI to identify residual risk that remains unmitigated by ongoing activities (e.g., software assurance, hardware assurance, AT, supply-chain risk management, and RMF). This approach allows programs to effectively manage program risk and track progress on the mitigation of these risks.



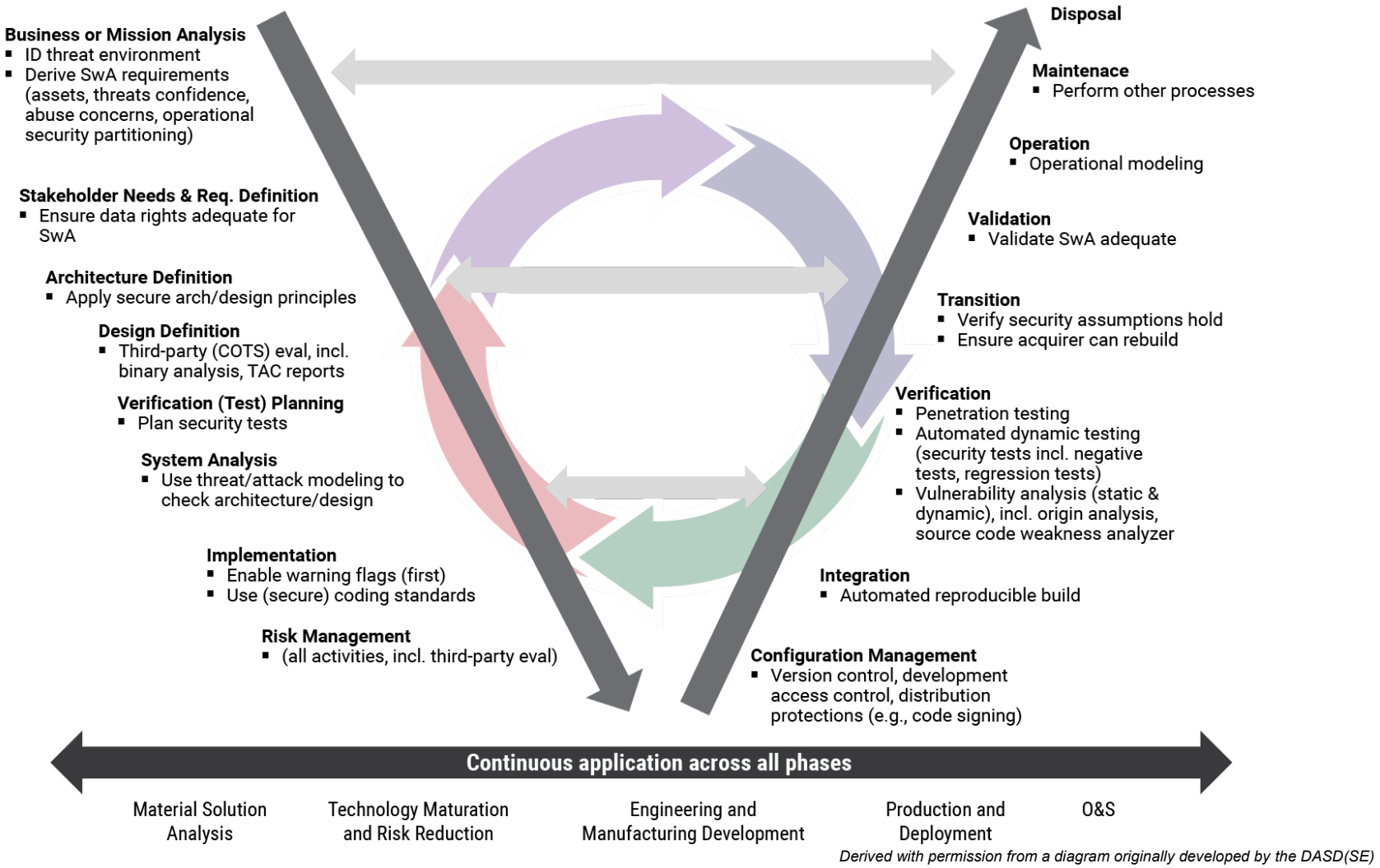
Derived with permission from a diagram originally developed by the DASD(SE)

Figure 5: Conceptual View of the Cyber Integrator

5.5 Software Engineer

The software engineer is responsible for showing how software assurance activities are addressed in each system lifecycle phase and the associated work products. A representative set of activities is shown in Figure 6. Although the software engineering team does not typically get involved in performing all these activities, its role is crucial in addressing all identified vulnerabilities and reducing the overall risks. As noted in the system engineering V-model, several of these activities may be repeated at different points in the lifecycle.

The software engineering development team, in close coordination with system engineering, develops the Software Development Plan (SDP), which is centered on the software development lifecycle and the associated methods, processes, work products, and tools. A cybersecurity or software assurance engineer works with the software engineering development team to ensure that appropriate software assurance practices and tools are integrated into the software development lifecycle. If program contractors are responsible for the SDP, the software engineering development team must specify the associated requirements for the SDP. Many established software development lifecycles and management techniques are being used in the commercial and DoD sectors, such as the Waterfall Model, Adaptive Software Development (ASD), Scrum, Extreme Programming (XP), Rapid Application Development (RAD), the Rational Unified Process (RUP), Kanban, and Cleanroom Development.



Derived with permission from a diagram originally developed by the DASD(SE)

Figure 6: Software Assurance Activities During the Acquisition Lifecycle

One might assume that all software development processes or methodologies can be characterized as either agile or heavyweight; however, this assumption would be erroneous. The reality is that each development approach has its own degree of agility. Generally, lightweight methodologies are less document oriented and more code oriented, meaning that the team considers the source code to be a key component of the project documentation. Regardless of the software development process selected, the SDP should contain adequate software assurance activities based on mission objectives and success criteria and ensure that software assurance is engineered into each phase of the development cycle.

5.5.1 Cybersecurity or Software Assurance Engineer

Cybersecurity or software assurance engineers understand the ways in which software can be vulnerable and the available practices and tools that can be used to reduce this risk. Cybersecurity engineers are tasked to address DoD 8510.01, RMF for DoD IT, which the policy defines as follows [DoD 2016a, Incorporating Change 2]:

> *Prevention of damage to, protection of, and restoration of computers, electronic communications systems, electronic communications services, wire communication, and electronic communication, including information contained therein, to ensure its availability, integrity, authentication, confidentiality, and nonrepudiation.*

This policy applies to "all DoD IT that receive, process, store, display, or transmit DoD information." This policy is now interpreted to include any technology that uses a DoD network.

The software assurance engineer is tasked with addressing the software vulnerability aspects of DoD 5000.02, which requires that "Software assurance vulnerabilities and risk-based remediation strategies will be assessed, planned for, and included in the Program Protection Plan (PPP)" [DoD 2017, Incorporating Change 3].

Both cybersecurity and software assurance engineering roles bring specialized knowledge and experience about the ways in which software can be vulnerable and the practices and tools available to identify and remove vulnerabilities. The primary responsibility of these roles is to apply this specialized knowledge and experience to the system at hand by working with the other engineers and stakeholders to integrate the most applicable practices and tools in the development lifecycle and to ensure that appropriate actions are taken to mitigate risk to the system, both during development and post-deployment.

5.6 System Security Working Group

Responsibilities for ensuring that software functions as intended and levels of risk are acceptable extend beyond the PMO. Although not a panacea, it is highly recommended—given the complexity, functional relationships, and deliverables associated with software assurance and cybersecurity—that the PMO establish and charter a System Security Working Group (SSWG) with management responsibility for all of these related functions within the PMO. The SSWG may also be referred to as a Program Protection Integrated Product Team or a System Security Engineering Working Group. This is not to imply that system and software security risks should be managed differently from other program risks once they are identified and qualified.

The SSWG should be chaired by someone charged with coordinating software assurance and cybersecurity, such as the systems security engineer or the CI. The core government SSWG membership includes representatives such as the information system security manager (ISSM), systems security engineer, software engineer lead, software assurance engineer, cybersecurity engineer, logistician, program security manager, and representatives from AT, cyber test and evaluation (T&E), users, contracting, and intelligence. The SSWG brings in additional subject-matter experts from outside the PMO as needed to address specific agenda items. Those experts might include representatives from acquisition, budget/finance, and other functional areas. The government membership is often mirrored by non-voting prime and subcontractor membership for most meetings.

A principal function of the SSWG team is to ensure that all software assurance and cybersecurity management processes and technical approaches are aligned with the system and software engineering development process, often presented for ease of understanding in terms of a V-model. The SSWG is also concerned with the enforcement of the DoD unlimited data/software rights, because without access to the data, the DoD's ability to assure that the software is free of vulnerabilities throughout the lifecycle is extremely hindered. Ultimately, a rigorous software assurance program and good SSE are key to meeting the system survivability KPP.

The SSWG relies on products from the acquisition and development lifecycle to perform its role. Input products include the Capability Development Document (CDD), acquisition strategy, design documentation, DoD instructions, source code with all associated build components, and other products as needed to accomplish its mission. The output products developed by the SSWG can include the threat models, software-assurance-related system requirements and design constraints, PPP, cybersecurity strategy, Clinger–Cohen Act compliance documentation, test and evaluation master plan, and RMF documentation including the security plan, Risk Assessment Report, Plan of Action and Milestones (POA&M), and Security Assessment Plan.

The SSWG can also provide software assurance and cybersecurity inputs to other acquisition documentation such as the SEP, acquisition strategy, CDD, SDP, Lifecycle Sustainment Plan (LCSP), and Software Assurance Plan.

The SSWG's most important activities should include providing design guidance, design reviews, test results reviews, and recommendations for system progression that ensure appropriate software assurance and cybersecurity through the acquisition lifecycle at appropriate design events and milestones. The SSWG can provide the technical area experts in software assurance and SSE that the PM can rely on.

5.6.1 Information System Security Manager and Officer

The information system security manager (ISSM) is responsible for maintaining the day-to-day security posture and continuous monitoring of a system. The information system security officer (ISSO) is responsible for the overall IA of a program, organization, system, or enclave.

Assurance and trustworthiness are critical to supporting the warfighter and maintaining national security. The ISSM and ISSO play key roles in attaining and sustaining a system's ATO through the RMF process. The PMO expects the systems security engineer, the cybersecurity engineer,

and the CI to interface with the ISSM and ISSO. The ISSM should be invited to participate in the SSWG if it is chartered.

5.6.2 Intelligence Liaison Office

The Intelligence Liaison Office performs research to define threats to the program's system at both general and specific levels. Threats include malicious activities up to and including sources at Adversary Threat Tier 4 (ATT 4). These include nation-state-level, network-based attacks against program subsystems that are connected to the DoDIN. ATT 4 adversary attacks against those portions of the system may be designed to either exfiltrate classified data from the system or affect the integrity or availability of the software and information in the system. ATT 4 adversary attacks are likely to be directed against the weapon system and SCADA elements in the system (e.g., supervisory computers, remote terminal units, or programmable logic controllers).

Based on gathered intelligence, the Intelligence Liaison Office may be able to show that an adversary could attempt to exploit weaknesses in DoD software. The consequence of an attack on the program's software will manifest itself in terms of confidentiality, integrity, or availability issues and could potentially impact the mission of the system or cause some cyber-physical effect. The Intelligence Liaison Office works with the SSE to review the National Institute of Standards and Technology (NIST) database for exploits and the NVD to ensure that the program's threat and weakness picture is as complete as possible. This information informs the program's threat modeling efforts, which are used to derive the system's software assurance requirements and design constraints. The Intelligence Liaison Office would be a resource to the SSWG if it is chartered.

5.6.3 User Community

In support of the user community, a survivability KPP is mandatory for all manned systems and is designed to enhance personnel survivability in an asymmetric threat environment (DoD 2015d, Enclosure B, Appendix D1). The attributes for the survivability KPP are identified in the CDD and Capability Production Document (CPD) and apply to all pre-Milestone C programs.

The measurement of success in meeting the user requirement for survivability in a cyber-contested environment is usually documented in the classified CDD. The CDD will contain threshold and objective requirements for both platform survivability and mission survivability. The user community felt it was necessary to break these into two different requirements because of the criticality differences in losing a weapon system platform due to a cyber attack and the failure of a single mission due to a cyber attack. The PMO should incorporate such requirements as appropriate into the contracted requirements for an acquisition to drive the acquisition and development lifecycle and establish an appropriate level of concern for software assurance and cybersecurity risk.

Table 2 shows the 10 Cyber-Survivability Attributes (CSAs). Note that software assurance has a large role to play in each of these CSAs. If the program cannot provide for software integrity in all phases of the system development and operational lifecycle, it will fail one or more of these CSAs, resulting in failure of the system to survive in a cyber-contested environment at the threshold levels needed by the user community.

Table 2: System Survivability (SS)/KPP Pillars Cyber-Survivability Attributes

SS KPP Pillars	Cyber-Survivability Attributes (CSAs)	
Prevent	CSA 01 - Control Access	CSA 10 - Actively Manage Systems Configuration to Counter Vulnerabilities at Tactically Relevant Speeds
	CSA 02 - Reduce Cyber Detectability	
	CSA 03 - Secure Transmissions and Communications	
	CSA 04 - Protect Information from Exploitation	
	CSA 05 - Partition and Ensure Critical Functions at Mission Completion Performance Levels	
	CSA 06 - Minimize and Harden Cyber Attack Surfaces	
Mitigate	CSA 07 - Baseline & Monitor Systems, and Detect Anomalies	
	CSA 08 - Manage System Performance if Degraded by Cyber Events	
Recover/Resiliency	CSA 09 - Recover System Capabilities	

Reprinted with permission from the Defense Acquisition University [Wilson 2017]

5.6.4 Program Security Manager

The program security manager is nominally responsible for generating several documents supported by other members of the SSWG and ensuring a secure environment for the program. Other responsibilities include the following:

1. generating the Security Classification Guide as part of the PPP in accordance with DoD Manual 5200.01 [DoD 2018c]
2. generating the operations security (OPSEC) plan in accordance with DoD Directive 5205.02 [DoD 2018b]. The focus is on securing the development environment. The OPSEC plan should be referenced by the SDP and has a direct impact on the software assurance effort.
3. managing the personnel security (PERSEC) program to ensure that granting an individual access to classified matter on a program will not endanger the common defense and security and would be consistent with national security interests
4. ensuring the program follows the guidance provided in NIST Special Publication (SP) 800-100: *Information Security Handbook: A Guide for Managers* [NIST 2007]. By incorporating major elements from SP 800-100 when establishing and implementing an information security program, the program is supporting software assurance.
5. ensuring the program office establishes a rigorous communications security (COMSEC) program to protect both classified and unclassified traffic on military communications networks, including voice, video, and data
6. ensuring compliance with the FAR requirement for incorporation of a DD Form 254 in the contract, if the contract with the prime includes classified materials. DD Form 254 provides to the contractor the security requirements and the classification guidance that are necessary to perform on a classified contract.

Both cybersecurity and software assurance engineers rely heavily on the documents produced and practices enforced by the program security manager as a fundamental foundation to minimizing risk to the system.

5.6.5 Program Logistics/Integrated Product Support Team

The program logistics management team works closely with the program team, including the software development team, to be sure that software assurance is included in the 12 Integrated Product Support (IPS) elements in accordance with the *Product Support Manager Guidebook* [DoD 2016b].

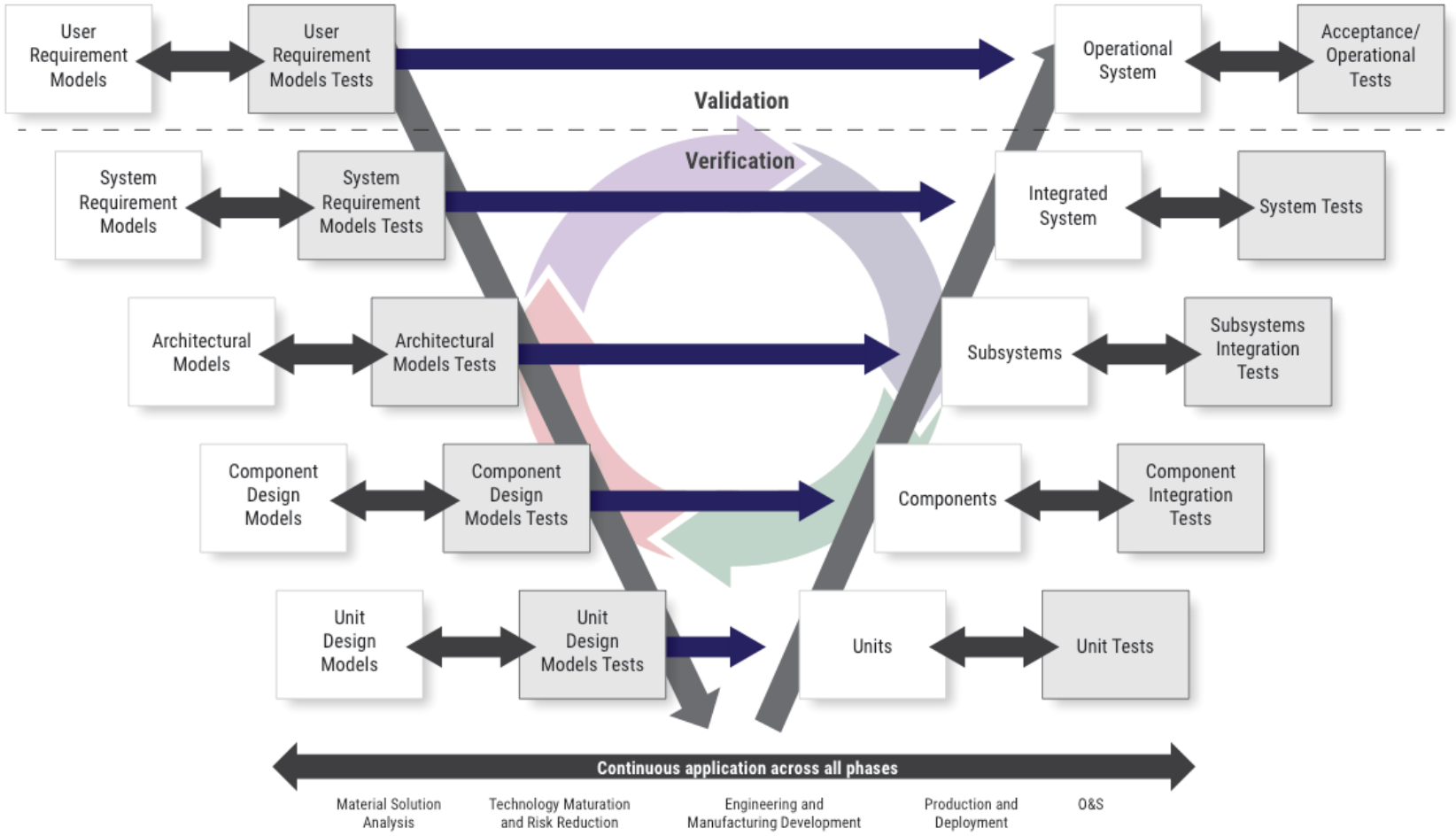
5.6.6 Test and Evaluation Team

A PM must be aware that DoD systems increasingly depend on complex, interconnected embedded and information technology environments. These environments are inherently vulnerable, providing opportunities for adversaries to compromise systems and negatively impact DoD missions. Potential software assurance vulnerabilities, when combined with a determined and capable threat, pose a significant security problem for the DoD and its warfighters. Software assurance T&E assists in the development (i.e., developmental test and evaluation [DT&E]) and fielding (i.e., operational test and evaluation [OT&E]) of systems to address this problem. The PMO would expect the systems security engineer, the cybersecurity engineer, and the CI to interface with these resources. The T&E team should be invited to participate in the SSWG if it is chartered.

As depicted in Figure 7, T&E is the process by which a system or components are compared against requirements and specifications through testing. The results are evaluated to assess progress of design, performance, supportability, and other attributes. DT&E is an engineering tool used to reduce risk throughout the acquisition lifecycle. OT&E is the actual or simulated employment, by typical users, of a system under realistic operational conditions.

Cybersecurity test and evaluation has six phases across the acquisition lifecycle, as described in the *Cybersecurity Test and Evaluation Guidebook* [DoD 2018a]. These phases are described below and shown in Figure 8. Cybersecurity test and evaluation phases are iterative; activities may be repeated several times due to changes in the system architecture, new and emerging threats, or system environment. Any of these actions can impact software assurance, requiring coordination with the PMO resources responsible for software and assurance risk. The first four phases support DT&E while the remaining two phases support OT&E.

1. Cybersecurity requirements gathering: The PM should seek advice from the SSWG as input.
2. Cyber-attack surface characterization: The attack surface is the system's exposure to reachable and exploitable vulnerabilities, such as hardware, software, connections, data exchange, and removable media that might expose the system to potential threat access. System engineers and system contractors identify these vulnerabilities through threat modeling as well as CPI and critical components documented in the PPP.
3. Cooperative vulnerability identification: The purpose of this phase is to identify vulnerabilities that may be fed back to system designers by the developers and component testers.
4. Adversarial cybersecurity: This phase includes an evaluation of the system's cybersecurity in a mission context, using realistic threat exploitation techniques in a representative operating environment.



Derived with permission from a diagram originally developed by the Defense Acquisition University [Wilson 2017]

Figure 7: Test & Evaluation Lifecycle Tests

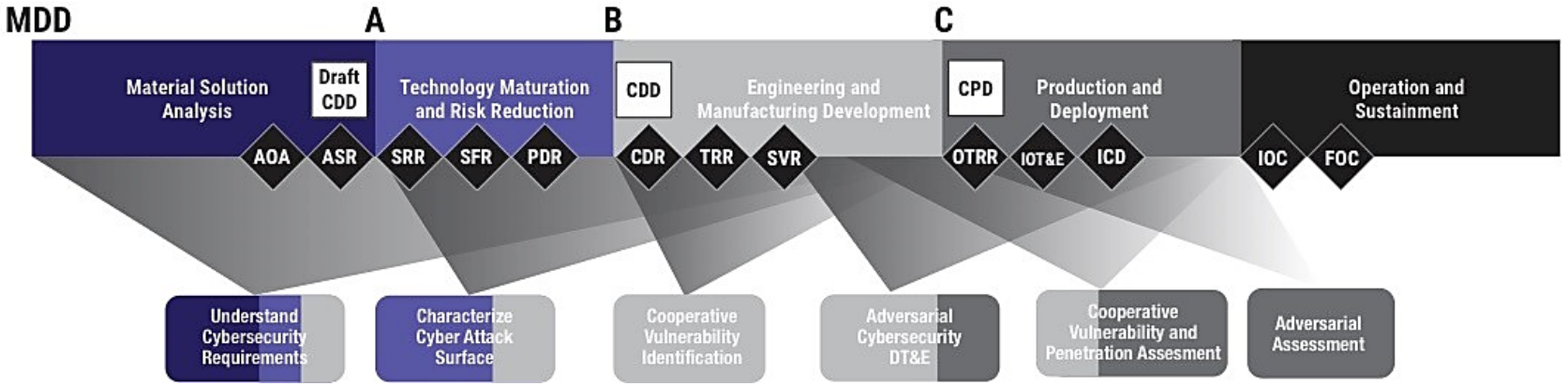
5. Cooperative vulnerability and penetration assessment: The Operational Test Agency (OTA) completes this phase either before or following Milestone C, as appropriate. The purpose of this phase is to provide a comprehensive characterization of the cybersecurity status of a system in a fully operational context. This operational test shall be conducted by a vulnerability assessment and penetration testing team. The assessment should be conducted in the intended operational environment with representative operators. This testing may be integrated with DT&E activities if conducted in a realistic operational environment and approved in advance by the Director of Operational Test and Evaluation.
6. Adversarial assessment: This phase assesses the ability of a unit equipped with a system to support its missions while withstanding validated and representative cyber threat activity. In addition to assessing the effect on mission execution, the OTA shall evaluate the ability to protect the system, detect threat activity, react to threat activity, and restore mission capability degraded or lost due to threat activity.

Table 3 identifies the roles of the different test and assessment teams. A National Security Agency (NSA)-certified red team is required for any adversarial event conducted on the DoDIN. Blue teams are usually qualified by the sponsoring agency.

Table 3: Roles of Different Test and Assessment Teams

Security Controls Assessment Team	Vulnerability Assessment (Blue Team)	Threat Representative Testing (Red Team)
Assess compliance with security controls	Comprehensive	Exploit one or more known or suspected weaknesses
Execute the Security Assessment Plan	Identifies any/all known vulnerabilities present in systems	Attention to specific problem or attack vector
Linked to the Security Assessment Report Activities	Reveals systemic weaknesses in security program	Develops an understanding of inherent weaknesses of system
Based on Security Technical Implementation Guides (STIGs) or similar documentation	Focused on adequacy and implementation of technical security controls and attributes	Both internal and external threats
Can be determined by multiple methods: hands-on testing, interviewing key personnel, etc.	Multiple methods used: hands-on testing, interviewing key personnel, or examining relevant artifacts	Model actions of a defined internal or external hostile entity
Include a review of operational and management security controls	Feedback to developers and system administrators for system remediation and mitigation	Report at the end of the testing
Conducted with full knowledge and assistance of systems administrators, owner, and developers	Conducted with full knowledge and cooperation of systems administrators	Conducted covertly with minimal staff knowledge
No harm to systems	No harm to systems	May harm systems and components and require cleanup

Reprinted with permission from the Defense Acquisition University [Wilson 2017]



Derived with permission from a diagram originally developed by the Defense Acquisition University [Wilson 2017]

Figure 8: The Six Phases of Cybersecurity Test and Evaluation

6 Software Assurance Throughout the Acquisition Lifecycle

What objectives should a PM focus on at each phase in the acquisition lifecycle?

This section focuses on software assurance objectives, activities, and work products in each phase of the Integrated Defense Acquisition, Technology, and Logistics Lifecycle Management System. This content is supported by the PM checklist in Appendix D.

This activities in this section are divided by DoD acquisition phase. For the treatment of software and software assurance, the processes that are performed in any one phase are repeatedly performed in almost all other phases since software is developed in each phase. As a result, there is some duplication. For example, there is an implicit assumption that engineering and manufacturing development (EMD) is just about coding and vulnerability testing. In reality, EMD includes software requirements development, design, code, test, and so forth—just as the previous and following phases do. As a result, each phase reflects activities based on the level of software assurance of all the software developed or acquired.

When applying software assurance to a more agile or iterative approach to system acquisition, the focus should be on achieving the intent of the various objectives listed regardless of phases. The software assurance processes are the same in all lifecycle approaches, but the way they are implemented is different. If an agile software development lifecycle is selected, the PM should understand the overlay of software assurance activities onto the agile development approach.

6.1 Materiel Solution Analysis (MSA) Phase

The overarching objective of the MSA is to establish the trade space of alternatives by identifying risks, appropriating lifecycle support, and estimating lifecycle costs.

Objective 1: Demonstrate Understanding of the Software Assurance Risks

Understanding risk is an objective for every phase in the acquisition lifecycle. Risk considerations need to extend beyond cost and schedule, and risks related to software assurance need to be included. It is assumed that there will be sufficient assembly of risk information from each step to influence appropriate action in subsequent steps.

Activities

- Perform an initial threat assessment to form a basis for deriving software assurance and cybersecurity requirements.
- Establish a requirements management system to document software assurance requirements.
- Document requirements elicitation and participation by stakeholders.
- Fund and document sufficient requirements analysis activities.
- Identify software assurance roles and responsibilities in the PPP.
- Incorporate software assurance requirements into the RFP.
- Train program personnel in the tools and processes of software assurance analysis and verification.

Work Products

- PPP
- software assurance requirements for the RFP
- software assurance training materials
- system threat models

Measurements and Evidence

- number of stakeholders and effort supporting requirements activities
- number of people trained to use specific tools and techniques
- number of software-assurance-related requirements
- software assurance requirement change counts and rate

Objective 2: Develop a Baseline Lifecycle Software Assurance Strategy

Activities

- Develop a software assurance strategy encompassing the system lifecycle.
- Record criticality and threat analysis in the PPP.
- Identify requirements to mitigate software vulnerabilities, defects, or failures based on mission risk.

Work Products

- software assurance strategy encompassing the system lifecycle
- criticality and threat analysis in the PPP
- requirements to mitigate software vulnerabilities, defects, or failures based on mission risk

Measurements and Evidence

- initial plans for software assurance activities throughout the lifecycle
- estimates of software assurance costs throughout the lifecycle
- numbers of requirements by category
- estimates of software assurance risk exposure

6.2 Technology Maturation and Risk Reduction (TMRR) Phase

At this point, it is assumed that the initial set of requirements has been defined for the system of interest and that the requirements have been integrated and approved.

Objective 1: Make Sound Architectural Decisions

Activities

- Establish misuse and abuse cases (based on threat expectations) that will be addressed by the architecture.
- Establish and review baseline architecture for weaknesses (CWE) and attack exposure (Common Attack Pattern Enumeration and Classification [CAPEC]), identify attack surface, and determine mission impact.
  - Many of the CWEs are at the implementation level, not the design level, but a program can at least identify countermeasures early in the lifecycle.
  - Decisions for preventing, mitigating, and recovering/resisting can be made for identified weaknesses.
- Review options for programming language, architecture, and operational environments.
- Identify options for software assurance tools and automation along with potential costs (see Appendix H).

Work Products

- misuse and abuse cases for detailed design and testing
- baseline architecture
- technical tradeoff documents of options for programming language, architecture, and operational environments
- software assurance tool options based on expected threats, tradeoffs, and potential costs

Measurements and Evidence

- software assurance expertise of participants
- number of misuse and abuse cases addressed
- staffing expertise and effort applied to architectural design
- size of the initial architecture, number of pages, number of components, and number of views
- records of architecture evaluation activities, for example, Architecture Tradeoff Analysis Method® (ATAM®) findings, total effort applied, and number of participants
- number of architecture evaluation activities that included software assurance
- software assurance requirements changes documented in the revision control system
- contribution of software assurance to the should-cost estimate
- changes to lifecycle cost estimates from the MSA phase

Objective 2: Enforce Secure Design Principles

Activities

- Ensure secure design principles.
  - There are several design principles that could be used, such as system element isolation, least common mechanism, least privilege, fault isolation, and input checking and validation.
- Verify and enforce principles at structured design reviews (see Appendix E).

Work Products

- documents reflecting application of secure design principles

Measurements and Evidence

- design document size
- design criteria and design review checklists
- design review issues
- design changes after baseline of items for review

6.3 Engineering and Manufacturing Development (EMD) Phase

Objective 1: Enforce Secure Coding Practices to Remove Potential Software Vulnerabilities

Activities

- Implement a software assurance process for development, including configuration control, tools, and automation.
- Perform code inspection and static analysis.
  - There are several approaches to include, such as secure coding rules and guidelines, automated code-level vulnerability scans (CWE), COTS library and component scans (CVE), and defect tracking.
- Address identified coding violations and CWEs.

Work Products

- code inspection and static analysis materials and measures

Measurements and Evidence

- inspection meeting logs that show what artifacts were inspected, who performed the inspections, and their qualifications
- inspection records showing effort applied to the inspections and product size
- inspection checklists
- logs of inspection or static analysis findings
- logs of issues evaluated, issues addressed, and remaining risks
- developer effort remediating inspection and static analysis findings

- description of where inspection and static analysis are applied in the development workflow
- description of static analysis tools used and known effectiveness

Objective 2: Apply Static Binary/Bytecode Analysis

These activities are required when data rights were not properly exercised or other limitations exist. Otherwise, the analysis should be primarily on the source code.

Activities

- Binary analysis can detect weaknesses within the binary objects and in the final executable code.
- Weaknesses and vulnerabilities can be uncovered in supporting binary/bytecode libraries.

Measurements and Evidence

- documentation that analysis is integrated into the build workflow
- number of issues and density of issues found
- number of issues addressed and remaining risk
- issue logs showing the weaknesses and vulnerabilities found along with prioritization and disposition
- developer effort remediating issues identified during the static analysis
- description of how the build system automatically executes the static analysis
- description of the specific static analysis tools applied and their known efficacy
- density of weaknesses removed by the static analysis (in the context of other assurance activities)

Objective 3: Use Rigorous Vulnerability Testing

Activities

- Perform non-functional testing.
  - There are several approaches to include, such as fuzz testing, sandbox testing, penetration testing, test (code) coverage metrics, and branch points.
  - Develop and execute test cases as informed by threat modeling and misuse and abuse cases, and assign to testing events.
  - Include in the regression suite tests some negative test cases that should fail if security mechanisms work properly, for example, using a seven-character password on a system that requires at least eight-character passwords.
- Review and monitor the CAPEC list of software weaknesses.

Work Products

- documents showing accomplishment of vulnerability testing and measures collected
- regression test case suite

Measurements and Evidence

- types of testing applied
- number of software assurance test cases prepared and the number actually executed
- number of negative test cases
- ratio of test cases per requirement and product size
- number of test failures
- test case line-of-code coverage and branch coverage
- change orders triggered by software assurance testing
- specific weaknesses and vulnerabilities tested
- issue logs showing weaknesses and vulnerabilities found along with their prioritization and disposition

Objective 4: Ensure Software Weaknesses and Vulnerabilities Are Not Checked into Version Control

Activities

- Version control, revision control, and configuration management should be applied to all source code, libraries, and test cases.
- All detected weaknesses and vulnerabilities should be prioritized and tracked to completion using an issue tracking system.
- All detectable weaknesses and vulnerabilities should be mitigated before source code is checked into the revision control system.
- Regression testing should be confirmed to ensure that previously addressed issues are not re-introduced.

Work Products

- manifests of content for specific product releases
- issue tracking logs and test verification logs

Measures and Evidence

- All detected weaknesses and vulnerabilities can include a prioritization and grouping to demonstrate they were analyzed, for example, a severity measure such as the Common Vulnerability Scoring System, a CWE grouping, or a Defense Information Systems Agency (DISA) Continuous Monitoring and Risk Scoring dashboard.
- Issue logs should include references to weaknesses and vulnerabilities along with the disposition.
- Automated build systems can verify remediation by requiring passing regression tests or scans prior to source code commits.

6.4 Production and Deployment (PD) Phase

A PM should not assume that software development has been completed by the start of the PD phase. Software for a system is almost always in development in every phase of the acquisition lifecycle. For example, there will probably be new requirements, requirements changes, or design changes that require software development. Thus, the PD phase, just like every other phase, includes software requirements development, design, code, test, and so forth.

Integration can include components that are COTS, GOTS, vendor provided (which will include software from their supply chain), organic development, and legacy. Software assurance in PD must confirm that the risk from all sources has been appropriately addressed to provide the required assurance.

Objective 1: Establish Integration Assurances

Activities

- Secure the integrated production system.
- Define and maintain the regimen of automated regression testing.
- Address software assurance test cases assigned to integration testing.
- Establish the chain of custody from development to sustainment for all sources.
- Confirm configuration management of source code links with the chain of custody.
- Ensure test coverage of all functions and components.

Work Products

- test documentation (tests assigned, tests executed, and results)
- configuration manifest
- integrated build verification that includes all components

Measurements and Evidence

- documentation of automated regression test
- number of test cases and test cases automated
- number of software assurance test cases performed and automated
- regression test execution logs
- mapping of test coverage of software assurance requirements
- software surface addressed in integration testing
- number of changes triggered by test results

Objective 2: Establish Deployment Assurances

Activities

- Confirm receipt of software from all sources through chain of custody.
- Develop handoff of ownership for software assurance countermeasures in the PPP.
- Transfer build environments, test suites, and scanning tools.
- Transfer coding rules and guidelines.

Work Products

- documentation showing the content of the handoff
- documentation showing unaddressed vulnerabilities (residual risk)

Measurements and Evidence

- list of secure deployment assurances, such as digital signature, AT, and so forth
- evidence of deployment scripts
- evidence of build system transfer
- evidence of unaddressed vulnerabilities

6.5 Operation and Sustainment (O&S) Phase

A PM should assume that system software is almost always in development in the O&S phase of the acquisition lifecycle. For example, there will usually be software development to support new capabilities and other sustainment activities. Thus, the O&S phase, just like every other phase, includes software requirements development, design, code, test, and so forth.

Objective 1: Monitor Threats and Attacks

Activities

- Perform security activities in operation.
  - Several approaches can be taken, such as monitoring the system, tracking threats, detecting and reporting attacks and breaches, and maintaining awareness of the attack surface, mission impact, and other factors.

Work Products

- reports and procedures associated with security of operation

Measurements and Evidence

- plans that document monitoring tools and approaches
- analysis of logs from monitoring tools
- training records
- reports from activities, review, or changes to threat model or attack surface

Objective 2: Identify and Fix Vulnerabilities for All Software

Identifying and fixing vulnerabilities should be accomplished as soon as possible, not in months or years. For example, a fix must be deployed before an adversary takes advantage of it, and this depends on the adversary's timeline, not on what is convenient to the program.

Activities

- Plan for monitoring and controlling residual risk (remaining vulnerabilities).
- Plan for monitoring and maintaining all vendor-managed software components (COTS, GOTS, open source, firmware, code libraries, etc.).
- Increase assurance in software maintenance.
  - Several approaches can be taken, such as training programmers in practices and tools for secure coding, vulnerability detection, code scanners, and structured reviews; finding vulnerabilities and eliminating them; and maintaining and strengthening software assurance regimens from development and test.

Work Products

- training documentation and vulnerability reports
- upgrade and update plans for software products

Measurements and Evidence

- plans for applying vulnerability detecting tools
- plans that show adequate staffing level and experience to identify and fix vulnerabilities
- issue tracking logs that reference weaknesses and vulnerabilities and their disposition
- revision control logs that reference changes from issue tracking
- analysis of logs from network scanners, virus and vulnerability scanners, and so forth
- analysis of logs from origin analysis or other static scan

Objective 3: Perform Software Assurance Awareness and Update Activities

Activities

- Update the threat model.
- Raise the software assurance posture.
- Monitor updates to the PPP.
- Mitigate residual weaknesses and vulnerabilities.
- Evolve application-based attack detection, mitigation, and recovery.
- Adapt to mission evolution.
- Adapt to technology evolution.

Work Products

- materials that raise the software assurance posture
- responses to updated threat model
- responses to changes in threat model

Measurements and Evidence

- plans and reports from requirements and design activities that reference software assurance enhancements
- updates to the PPP
- requirements changes that reference software assurance enhancements
- issue tracking and revision control logs that reference changes for software assurance enhancement upgrade, including enhancements and mitigation of residual weaknesses
- work plan that includes software assurance enhancement activities
- logs from static analysis that show reductions of vulnerabilities to acceptable levels over time
- revision control of build system that references changes to libraries or updates to software assurance tools

7 Software Assurance Results for Technical Reviews

How should a PM measure software assurance progress at major milestones?

This section focuses on software assurance results that should be delivered to support the technical reviews at major milestones in each phase of the Integrated Defense Acquisition, Technology, and Logistics Lifecycle Management System. This content is supported by the PM checklist in Appendix E.

7.1 Software Assurance Results for the Acquisition Strategy Review (ASR)

- software assurance roles and responsibilities that have been assigned
- selection of secure design and coding standards demonstrated
- identification of functions that use software
- identification of software assurance activities across the lifecycle
- identification and mitigation plan for vulnerabilities, defects, and failures
  - Establish requirements to mitigate software vulnerabilities, defects, or failures based on mission risks.
- software assurance requirements incorporated into solicitations
- plans established and funded for software assurance training and education
- attack knowledge that has been analyzed and documented indicating how the system may be attacked
  - This review should include attack patterns and threat modeling.
- plan for static analysis and other automated verification procedures

7.2 Software Assurance Results for the System Requirements Review (SRR)

- automated tools considered and rationale (cost, availability, threat coverage, usage experience on the team, etc.) for selection of design and vulnerability scan and analysis (see Appendix H)
- software assurance requirements defined for all software components
  - Programming languages, architectures, development environments, and operational environments all have security requirements.
- plans to address software assurance in legacy code
- software assurance requirements defined for faults and attacks
- assurance requirements established for software to deter, detect, react, and recover from faults and attacks based on threat modeling

- documented results for sufficiency of software assurance as part of reviewing, inspecting, and tracking software and requirements
  - results from initial software assurance reviews and inspections, and established tracking processes for completion of assurance requirements

7.3 Software Assurance Results for the System Functional Review (SFR)

- sufficiency of software assurance requirements included in system requirements
- confirmed plans and sufficiency criteria to be used to establish a baseline architecture and review for weaknesses
  - Include CWEs and susceptibility to attack (CAPEC), and refine potential attack surfaces and mission impacts.

7.4 Software Assurance Results for the Preliminary Design Review (PDR)

- architecture and design results that demonstrate software assurance requirements and verified secure design principles have been addressed
- confirmation of document reviews and inspections
  - Determine whether initial software assurance reviews and inspections received from assurance testing activities are documented.
- confirmation that software assurance requirements map to module test cases and the final acceptance test cases
- confirmation that automated regression testing procedures and tools are defined and implemented
  - Automated regression testing procedures and tools should become part of the core process.
  - Verify sufficiency of the planned process for software assurance tools and automation.

7.5 Software Assurance Results for the Critical Design Review (CDR)

- evidence that secure coding practices are established and applied
  - Employ code inspection augmented by automated static analysis tools.
- evidence that implemented practices detect software vulnerabilities, weaknesses, and defects in the software; prioritize them; and remediate them as soon as possible
- evidence that assured chain of custody from development to sustainment has been developed and implemented for any known vulnerabilities and weaknesses remaining and mitigations planned
- evidence that hash checking is implemented for delivered products
- confirmation that processes are implemented for remediation of vulnerabilities
- evidence that processes are implemented for timely remediation of known vulnerabilities (e.g., CVEs) in fielded COTS components

- evidence confirming that planned software assurance testing provides variation in testing parameters
  - for example, applying test coverage analyzers
- evidence that program software with critical functions and critical components receives rigorous test coverage

7.6 Software Assurance Evidence for Activities After the CDR

Post-CDR events include software assurance activities such as system verification review, functional configuration audit, and production readiness review. Evidence should be provided that confirms the activities listed below are sufficiently addressed.

- Verify test resources and test cases, scenarios, and data.
- Continue to enforce secure design and coding practices through inspections and automated scans for vulnerabilities and weaknesses.
- Automate vulnerability discovery, and execute remediation (see Appendix H).
  - Maintain automated code vulnerability scans, reporting, and prioritization, and execute defect remediation plans.
- Maintain and enhance automated regression tests, and employ test coverage analyzers to increase test coverage.
- Periodically during development, conduct penetration tests using the enhanced automated test coverage procedures established during previous activities.
- Monitor and respond to threats and attacks.
  - Monitor evolving threats and attacks, respond to incidents and defects, identify and fix vulnerabilities, and incorporate software assurance-enhancing upgrades.
  - The PMO should provide a plan for updates, replacements, maintenance, or disposal of CPI, components, and software with mission functions.
- Ensure the chain of custody from development to sustainment, and during sustainment, for the record of weaknesses and vulnerabilities remaining and mitigations planned.

8 Measuring, Budgeting, Scheduling, and Controlling Software Assurance Products and Processes

How do PMs plan and control program activities to achieve the program's software assurance objective?

This section provides guidance on how a PM, through the use of metrics and other sources, confirms the adequacy of in-place systems and software engineering work products, security controls, policies, and procedures. It also provides an approach to help PMs decide where to invest in additional security protection resources or identify and evaluate nonproductive activities and controls.

Technical risk management is a fundamental program management and engineering process that should be used to detect and mitigate vulnerabilities, defects, and weaknesses in software and hardware so they do not become breachable cyber vulnerabilities in deployed systems. The PM must establish responsibility for these within the engineering resources assigned to the program.

Cyber vulnerabilities provide potential exploitation points for adversaries to disrupt mission success by stealing, altering, or destroying system functionality, information, or technology. PMs describe in their PPPs the program's CPI and mission functions and components, the threats to and vulnerabilities of these items, and the plan to apply countermeasures to mitigate or remediate associated risks. Software is typically predominant in system functionality, and software assurance is one of several countermeasures that should be used in an integrated approach that also includes information safeguarding; designed-in system protections; "defense-in-depth" and layered security; supply-chain risk management (SCRM) and assurance; hardware assurance; anti-counterfeit practices; AT; and program security-related activities such as information security, operations security, personnel security, physical security, and industrial security. Software assurance vulnerabilities and risk-based remediation strategies are assessed, planned for, and included in the PPP from a time frame early enough that resources can be planned and obtained.

8.1 Software Assurance, the NIST Risk Management Framework, and Continuous Risk Monitoring

Software is the foundation of systems comprising our nation's military power. The primary and important mission capabilities of all current and foreseen weapons systems are implemented in software. Software assurance is critical to the expanded use of software such that weapon system software is free of detectable vulnerabilities, defects, and weaknesses that could disrupt a mission or prevent its achievement. The elimination of vulnerabilities through the application of software assurance within the engineering of a system is needed in conjunction with the selection and implementation of security controls imposed through the NIST RMF. Both approaches are needed to reduce or eliminate the impact of vulnerabilities in software-enabled defense systems.

DoD policy mandates that the cybersecurity programs implemented by government agencies adhere to NIST standards and guidelines, specifically NIST SP 800-39, Managing Information Security Risk: Organization, Mission, and Information System View [NIST 2011]. This document outlines a risk management approach that evaluates risks in three tiers.

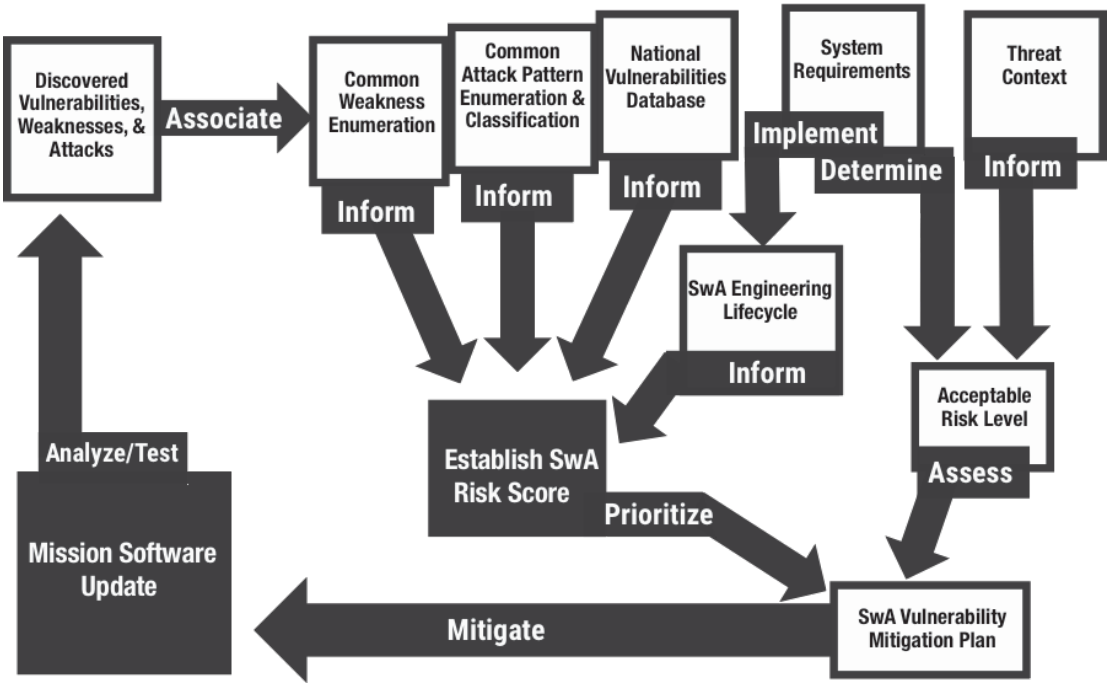
Tier 1 evaluates risks at the organizational level by establishing governance structures, defining priorities, developing investment strategies, and establishing common security controls for the enterprise. Tier 1 is the responsibility of the agency head and chief information officer.

Tier 2 evaluates risk at the mission/business level by defining processes and prioritizing those processes according to the guidance from Tier 1. Tier 2 is the responsibility of mission and business owners.

Tier 3 evaluates risk at the system level. Most of the detailed policy for cybersecurity risk management threats at the system level in Tier 3 comes from the NIST RMF. This framework is risk-based and focused on the success of the mission throughout the acquisition lifecycle by implementing security controls for information asset protection and verifying the results of activities related to selection and implementation of controls at different points in the acquisition lifecycle. Thus, the results are very important to the PM.

The PM's responsibility includes managing the continuous risk state of the system as it undergoes engineering development and sustainment while the program team makes tradeoffs relative to engineering-in software assurance activities. Eighty-four percent of successful cyber attacks are directly and specifically waged on vulnerabilities in the applications that achieve the system mission [Clark 2015]; thus, it is critical that the software is resilient to cyber attacks. The PM is responsible for ensuring that software assurance is engineered-in throughout the acquisition cycle so that the system is resilient to attack. Working with the engineers assigned to address software assurance, the PM ensures that software assurance tools, techniques, and a methodology to engineer-in assurance are used from the beginning of concept development and throughout the system lifecycle so that vulnerabilities are discovered and fixed at the earliest possible point, thus reducing performance risk and resources.

Figure 9 provides a dynamic view of what happens when new vulnerabilities, weaknesses, and attacks are identified and associated changes are made to the program's risk table, which in turn can help the PM and other members of the program team prioritize changes to the software to mitigate the risk.



Derived with permission from a diagram originally developed by the DASD(SE)

Figure 9: Software Assurance Risk Management

8.2 Measurement for Software Assurance

PMs can lack confidence in the assurance of their software-reliant systems unless they have established methods to measure, monitor, and control how the software functions. Furthermore, a PM can have trouble making connections among the detailed, disparate data available throughout the system lifecycle.

Measurement in software development includes measurement of process performance enactment, measures of the software products, and derived measures of the process outcomes. Process enactment can demonstrate that some activities were performed, but it cannot directly measure their effectiveness. Process outcomes can show what the development process found, but not what it did not find.

Software process enactment includes the activities performed and efforts applied. The planned activities should address aspects of the security risk during development. The planned effort helps prioritize the activities, and comparison to actual effort is an indicator that the activities were not performed superficially. Other measures include number of test cases, line-of-code coverage, path coverage, coverage for specific weaknesses of vulnerabilities, and time between the discovery of a defect or weakness and the closure of its remediation.

Product measures are sometimes the result of a process workflow activity. Outputs include documentation, code, warnings, test case failures, defects, and revision commits. Many of these should be easily countable if tools are included in an automated workflow. Defects, including weaknesses and vulnerabilities, should be counted. The density of defects, weaknesses, and vulnerabilities

should be normalized by product size measured in lines of code or function points (such as Common Software Measurement International Consortium [COSMIC] function points).

Process outcome measures typically include product measures at the exit of a process workflow activity. The relative results can be displayed by activity. For example, the fraction of weaknesses removed by each activity can be displayed as a line chart or Pareto diagram.

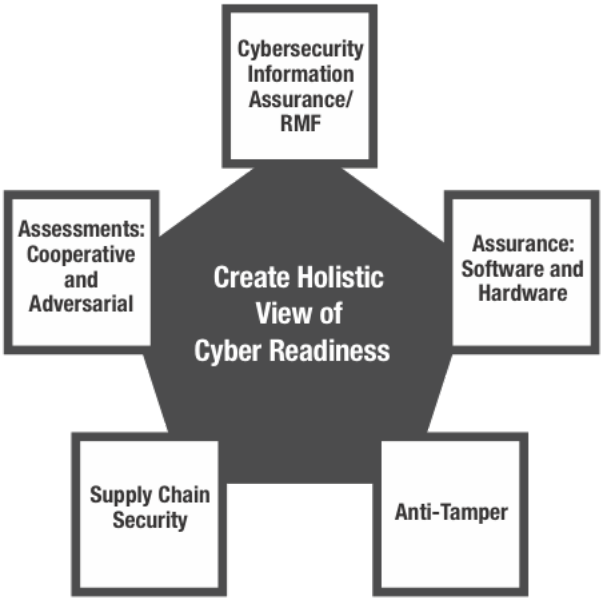
It is important to agree on what metrics, measures, and indicators will be reported in advance with the development teams so that they can arrange to collect and report the necessary data. They may also automate much of the data collection along with the workflow.

Several initiatives have been undertaken in the area of software assurance measurement. In 2013, Shoemaker and Mead produced a report that identified and described the current state of the practice in software assurance measurement [Shoemaker 2013]. The discussion focuses on the methods and technologies that apply to the domain of existing software products, software services, and software processes. The software assurance community of practice has an ongoing metrics effort. Furthermore, this guidebook highlights the software assurance activities in the Integrated Defense Acquisition, Technology, and Logistics Lifecycle Management System, which provide an exemplar of how software assurance objectives for each phase could be linked to metrics that offer evidence of how well the software engineering practices address software assurance objectives. While not yet a comprehensive list of all possible metrics, it provides a starting point for organizations to initiate security measurement by combining cybersecurity engineering and risk management.

Another noteworthy effort is being undertaken by the NSA Center for Assured Software, which is looking for properties of software that are indicators of its assurance level. For example, indicators might include the degree of confidence that software will securely and appropriately perform its intended functions, will not perform any unauthorized functions, and does not contain implementation flaws that could be exploited. The NSA has developed five activities to guide its effort:

1. Acceptance: Determine whether there are existing tools and techniques that address the software to be evaluated.
1. Extraction/inspection: Apply tools and techniques that extract relevant metadata from the software.
2. Analysis: Apply tools and techniques that query the metadata for properties or indicators of assurance.
3. Meta-analysis: Integrate output from multiple analytical tools and techniques to discern higher order assurance indicators.
4. Reporting: Transform analytical results into comprehensible reports.

Ultimately, a PM would like to have a measurement program that provides holistic cyber assessment and planning recommendations to meet software assurance and cybersecurity requirements for each phase of the acquisition lifecycle. A conceptual view is provided in Figure 10.



Derived with permission from a diagram originally developed by the DASD(SE)

Figure 10: Holistic View of Software Assurance/Cyber Readiness

8.3 Budgeting for Software Assurance

Software assurance budgets and spending are on the rise, with much of that spending going toward skills to support application security, intelligence and analytics, and data security, among other functions. The PM is often faced with the dilemma of mitigating a large set of software vulnerabilities during the system development lifecycle while operating with a limited set of resources with which to implement solutions. Selecting the appropriate software assurance solutions can be overwhelming. Furthermore, the challenge is exacerbated by the fact that software assurance strategies must consider the different perspectives of the program's stakeholders. One thing is clear: fixing software assurance defects earlier is cheaper.

Budgeting for software assurance is an evolving process and requires inputs from multiple DoD acquisition workforce disciplines. Cost estimating techniques need to be calibrated with respect to the characteristics of the system under development and the current knowledge of the software assurance environment. The ISSM, working with the PMO SSWG team, can provide an estimate of the budget needed to cover the lifecycle controls for software assurance from DISA, the RMF, and Information Assurance Vulnerability Alerts (IAVAs). The software representative will use established cost estimating techniques, adjusted to consider software assurance factors. The disciplines of T&E and integrated logistics support each provide methods of estimating software assurance costs. The program budget will normally be required to cover development and sustainment.

The Office of the Secretary of Defense (OSD) Office of Cost Assessment and Program Evaluation (CAPE) provides independent analysis and advice to DoD officials on matters of cost estimation

and cost analysis for weapons acquisition programs, including matters of program lifecycle cost. The OSD-CAPE is required to

- establish substantive guidance on the preparation of lifecycle cost estimates subject to review
- establish policy guidance on the cost and software data reporting system and monitor its implementation to ensure consistent and appropriate application throughout the DoD
- establish policy guidance on the Visibility and Management of Operating and Support Costs Program and monitor its implementation by each military department

The OSD-CAPE may also help establish a basis of estimate for cybersecurity and software assurance. In addition, the PM should require and validate a software assurance and cybersecurity plan from the prime contractor.

8.4 Scheduling for Software Assurance

Scheduling in project management often becomes a balancing act relative to software assurance and cybersecurity due to the large number of software assurance activities that occur over the lifecycle. Due to limited resources, the PM will probably be required to determine the level of confidence the program can obtain within the program constraints. A good source of advice for the PM within the program is the SSWG. Externally, the OSD-CAPE may be able to provide independent analysis and advice.

A principal function of the SSWG team is to ensure that all SSE management processes and technical approaches are aligned with the system engineering development process. Ideally, the SSWG can define the software assurance tasks such that they can be represented in the program's work breakdown structure. The next step is to sequence these tasks and place them into an Integrated Master Schedule (IMS) for the program. Once this step is completed, the SSWG can estimate the resources needed for each task and the duration of each task. If desired, it is possible at this point to combine the tasks into a schedule for just the software assurance tasks and determine the associated budget. The PM can then review the schedule and budget to determine the suitability relative to program objectives and constraints.

A number of standard and familiar tools and techniques are available to manage schedule development, such as Gantt charts, Program Evaluation and Review Technique (PERT) charts, and Microsoft Project. Scheduling, like most other management activities, requires monitoring and adjustment throughout the development lifecycle. Although PMs start out following the project plan's IMS, they need to monitor project performance against the estimates in the plan and adjust it as necessary. Changes to (almost) anything in the project will affect the schedule. The concept of the critical path is used to identify which activities affect the schedule directly and which can grow or shrink without affecting the overall progress.

Appendix A: Obtaining Additional Resources – Joint Federated Assurance Center

This appendix discusses the Joint Federated Assurance Center (JFAC) and describes ways that a PM can use it to obtain additional resources for managing software assurance in a software development or acquisition effort.

JFAC's Mission

The JFAC is a federation of DoD organizations that have a shared interest in promoting software and hardware assurance in defense acquisition programs, systems, and supporting activities. The JFAC member organizations and their technical service providers interact with program offices and other interested parties to provide software and hardware assurance expertise and support, including vulnerability assessment, detection, analysis, and remediation services as well as information about emerging threats and capabilities, software and hardware assessment tools and services, and best practices.

JFAC Portal

https://jfac.navy.mil/

Goals

- operationalize and institutionalize assurance capabilities
- organize to better leverage the DoD, interagency, and public/private-sector capabilities
- influence R&D investments and activities to improve assurance

Functions

- support PEOs and systems across the lifecycle
- sustain an inventory of software assurance and hardware assurance resources across the DoD
- coordinate an R&D agenda for assurance (software, hardware, firmware, systems, services, and mission)
- procure, manage, and enable access to enterprise licenses for vulnerability analysis and other tools
- communicate assurance expectations to broader communities of interest and practice

Objective: Reduce Program Risks and Costs

- grow DoD competency and practice in tools, techniques, and practices for software and hardware assurance
- resolve assurance issues through community collaboration, support, and best practices for remediation
- leverage commercial products, methods, and training

- provide practice-based guidance to tailor software and hardware assurance to program needs in contracts
- raise the bar for reduction of vulnerabilities and defects through spread of best practices
- heighten software assurance awareness through outreach, mentoring, training, education, and providing software assurance tools
- assess assurance capability gaps and recommend specific actions and products to close them

JFAC Reference Library Resources

- assurance policy and guidance
  - defense acquisition and system engineering
  - program protection and systems security
- engineering and systems assurance
- contract language
- JFAC toolbox
  - information on assurance tools and techniques
  - survey reports on use of enterprise software
- licenses
- reports and publications
  - whitepapers on Services and assurance topics
  - cybersecurity test and evaluation capabilities
- gaps
  - assurance metrics
  - penetration testing
  - assurance countermeasures
- best practices
- training
  - training materials from JFAC-sponsored events
  - assurance and cyber training resource links

Appendix B: Program Protection Plan, System Engineering Plan, and Related Documents

This appendix provides more detail about the Program Protection Plan and the System Engineering Plan.

Program Protection Plan (PPP)

The PPP provides a template and focal point for all security activities on a program. It is the single source document used to coordinate and integrate all protection efforts and is usually developed under the leadership of system engineering. Program protection is the integrating process for managing risks to advanced technology and mission system functionality from foreign collection, design vulnerability, supply-chain exploit/insertion, and battlefield loss throughout the acquisition lifecycle. The purpose of the PPP is to help programs ensure that they adequately protect their technology, components, and information. For example, the PPP is designed to deny access to CPI to anyone who is not authorized or who does not need to know and prevent inadvertent disclosure of leading-edge technology to foreign interests.

Metrics-based reporting should be used to track a program's progress against defined requirements and objectives. In addition, these reports should be analyzed to inform the risk-based scoring of weaknesses and vulnerabilities. Programs should use the risk score of individual vulnerabilities to determine priorities and mitigate risk as necessary. Reports should be captured in the PPP.

The PPP is developed by the PM, approved by the PM after an Initial Capabilities Document (ICD) has been validated, and submitted for approval per DoDI 5000.02 requirements. The Security Classification Guide is included as Attachment A to the PPP. The cybersecurity strategy, which also addresses software assurance, is included as Appendix E of the PPP. A draft is due for the development RFP release decision and is approved at Milestone B.

Software assurance is one of many system engineering activities, but it is an activity with an end goal: adequate security of the system. It is important to note that software assurance is a system engineering activity that focuses on making sure the system software functions as intended and is (relatively) free of vulnerabilities. Thus, software assurance is an important part of the PPP document. Some of the benefits that the PPP provides for software assurance include the following:

- focuses risk mitigation on the mission-enabling software functionality and components
  - describes countermeasures that target specific identified risks
  - applies the best program protection value for scarce resources
- raises visibility of program planning for software assurance
- promotes planning for software assurance activities
- influences program plans for software assurance

  - focuses planning on protecting the development process, the operational system, and the development environment
  - suggests potentially effective countermeasures
- helps programs tailor countermeasures to unique characteristics of the program
  - applies countermeasures in ways that make sense for the program's mix of COTS, GOTS, contractor, in-house, and legacy software
  - communicates utility/awareness and leverages analysis resources from the CVE, CWE, and CAPE
- baselines a process for continuous improvement
  - solicits feedback from programs
  - provides a mechanism for discovering and promoting specific best practices
- focuses on software assurance countermeasures that address malicious insertion
  - highlights how software assurance countermeasures are complemented by software quality assurance, IA, safety, and other activities
  - focuses on defeating an attacker with malicious intent
  - avoids malicious insertion threat—especially via the supply chain, development tools, software updates, software evolution, and bad actors

System Engineering Plan (SEP)

The plan for implementing countermeasures, KPPs, TPMs, and all supporting planning artifacts should be documented in the SEP and the PPP. The SEP is a living document that details the execution, management, and control of the technical aspects of an acquisition program from conception to disposal. The PM updates the SEP for each milestone review starting with Milestone A.

Failure to properly execute SSE activities is a risk and represents an important element of a program's overall risk. At any stage of the mitigation process, a program can use the CI to identify residual risk that remains unmitigated by ongoing SSE activities. This allows programs to effectively manage SSE risk and track progress on mitigating these risks.

In addition to the CI, there are other tools to help support the PM's understanding of the responsibilities of the workforce concerning cybersecurity. For example, a team at the Defense Acquisition University has developed the Cybersecurity and Acquisition Lifecycle Integration Tool (CALIT). CALIT went live in June 2016 and has been downloaded thousands of times by members of the defense acquisition workforce. CALIT was developed on the premise that effective integration of cybersecurity into the DoD acquisition lifecycle encompasses several different processes, including

- DoDI 5000.02, Operation of the Defense Acquisition System
- DoDI 8510.01, RMF for DoD Information Technology (IT)
- Cybersecurity Test and Evaluation
- Program Protection/SSE
- Cyber Threat Analysis

- DoDI 5200.39, Critical Program Information Identification and Protection Within Research, Development, Test, and Evaluation
- DoDI 5200.44, Protection of Mission Critical Functions to Achieve Trusted Systems and Networks

CALIT provides the user with insight into these supporting processes and the ability to visualize how these processes work together to promote cyber-resilient weapon systems.

Appendix C: Standards, Statutes, Regulations, and Guidebooks

This appendix provides a list of significant software assurance statutes, regulations, and guidelines that PMs should be familiar with. Enclosure 14 of DoDI 5000.02 provides policies and principles for cybersecurity in defense system acquisition and is the principal guidance for PMs and this guidebook.

Importance of System and Software Engineering Standards

PMs should know that using engineering standards in key technical disciplines—such as software engineering, system engineering, and software assurance—can enhance project performance and provide a common framework for communicating best practices for implementing effective system and software engineering activities on DoD acquisition projects. Consistent with Public Law 114-3283, Public Law 104-113.4, and Office of Management and Budget Circular A-119, the DoD encourages the adoption of voluntary consensus standards where practical, rather than developing new or updating existing government-unique specifications and standards.

Standards

The PM should be aware of two important standards that guide system engineering and software engineering: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE) Standards 15288 and 12207.

ISO/IEC/IEEE 15288, Systems and Software Engineering – System Life Cycle Processes, is a system engineering standard covering processes and lifecycle stages [ISO/IEC 2015]. This standard presents the system and software integration activities. Initial planning for the ISO/IEC 15288:2002(E) standard began in 1994 when the need for a common system engineering process framework was recognized. The first edition was issued in November 2002. In 2004 this standard was adopted as IEEE 15288. ISO/IEC 15288 was updated in February 2008 and May 2015. ISO/IEC 15288 is managed by ISO/IEC JTC1/SC7, which is the ISO committee responsible for developing ISO standards in the area of software and systems engineering.

ISO/IEC/IEEE 12207, Systems and Software Engineering – Software Life Cycle Processes, is an international standard for software lifecycle processes [ISO/IEC 2017]. This standard defines all the tasks required for developing and maintaining software. The ISO/IEC/IEEE 12207 standard establishes a lifecycle process for software, including processes and activities applied during the acquisition and configuration of the services of the system. There are 23 processes, and each process has a set of outcomes associated with it. In Annex E of the latest release of 12207 (ISO/IEC 12207:2017), the standard presents the software assurance process view. The purpose of this view is to provide objective evidence that the software achieves satisfactory levels of certainty that sufficient protection is achieved against intentional subversion or forced failure due to the software architecture, design, or construction of the code.

Statutes

The PM should be aware of the following statutes that guide system engineering and software engineering.

10 USC 2358 – Research and Development Projects

"The Secretary shall develop a coordinated plan for research and development on … computer software and wireless communication applications, including robust security and encryption…."

FY13 NDAA, Sec. 933

NDAA for Fiscal Year 2013, Section 932, Improvements in Assurance of Computer Software Procured by the Department of Defense, January 2, 2013 [P.L. 112-239 2013].

USD(AT&L), in coordination with the DoD CIO, "shall develop and implement a baseline software assurance policy for the entire lifecycle of covered systems. Such policy shall be included as part of the strategy for trusted defense systems of the Department of Defense."

Public Law 111-383

Ike Skelton National Defense Authorization Act (NDAA) for Fiscal Year 2011, Section 932, Strategy on Computer Software Assurance.

Public Law 112-239

NDAA for Fiscal Year 2013, Section 933, Improvements in Assurance of Computer Software Procured by the Department of Defense, January 2, 2013.

The NDAA requested (1) a research and development strategy to advance capabilities in software assurance and vulnerability detection, (2) state-of-the-art software assurance analysis and testing, and (3) an approach for how the DoD might hold contractors liable for software defects or vulnerabilities.

Public Law 113-66

NDAA for Fiscal Year 2014, Section 937, Joint Federated Centers for Trusted Defense Systems for the Department of Defense.

This law directed the DoD to establish a federation of capabilities to support trusted defense systems and ensure the security of software and hardware developed, acquired, maintained, and used by the DoD. This requirement led to the creation of the JFAC, which is managed by the DASD(SE).

Regulations

The PM should be aware of the following regulations that guide system engineering and software engineering.

5000.02 Operation of the Defense Acquisition System

Enclosure 14 of DoDI 5000.02 provides policies and principles for cybersecurity in defense system acquisition and is the principal guidance for PMs and this guidebook [DoD 2017].

The regulatory requirement for producing a PPP at Milestones A, B, and C and at the full-rate production/full deployment decision references DoDI 5200.39, which requires that PPPs address software assurance vulnerabilities and risk-based remediation strategies [DoD 2015b]. It also requires that PPPs include software assurance as part of vulnerability countermeasures.

Other Regulatory Documents

A large number of regulatory documents reference DODI 5000.02 as a source document. This section provides a sample set. Collectively these documents advocate program protection as the integrating process for managing risks to advanced technology and mission-critical system functionality from foreign collection, design vulnerability, supply-chain exploit/insertion, and battlefield loss throughout the acquisition lifecycle.

Software assurance spans the entire DoD Integrated Defense Acquisition, Technology, and Logistics Lifecycle, and system engineering ensures that software assurance is effectively architected and designed into the system. The guidance in the *Defense Acquisition Guidebook* (DAG) [DAU 2017], best practices contained in standards such as ISO/IEC/IEEE Standard 12207:2017 and ISO/IEC/IEEE 15288:2015, and industrial association products like the *National Defense Industrial Association (NDIA) Engineering for System Assurance Guide* inform the system engineering community [NDIA 2008].

In turn, a number of regulatory documents place controls on the development process by integrating security and risk management activities into the system development lifecycle. Examples include

- DoDI 8500.01, Cybersecurity [DoD 2014]
- DoDI 8510.01, Risk Management Framework (RMF) for DoD Information Technology (IT) [DoD 2016a]
- Committee on National Security Systems Instruction (CNSSI) 1253, Security Categorization and Control Selection for National Security Systems [CNSS 2014]
- NIST SP 800-53, Security and Privacy Controls for Federal Information Systems and Organizations [NIST 2015]

DoDI 5200.44

Protection of Mission Critical Functions to Achieve Trusted Systems and Networks (TSN), November 5, 2012 [DoD 2012]. Establishes policy and assigns responsibilities to minimize the risk

that DoD's warfighting mission capability will be impaired due to vulnerabilities in system design, sabotage, or subversion of a system's mission-critical functions or critical components.

DoDI 5200.39

Critical Program Information (CPI) Identification and Protection Within Research, Development, Test, and Evaluation (RDT&E), May 28, 2015 [DoD 2015b].

DoD Directive 5200.47E

Anti-Tamper (AT), September 4, 2015 [DoD 2015a].

USD(AT&L) Memorandum

Document Streamlining – Program Protection Plan (PPP), July 18, 2011 [USD(AT&L) 2011].

PM 15-001

Deputy Secretary of Defense Policy Memorandum (PM) 15-001, Joint Federated Assurance Center (JFAC) Charter, February 9, 2015 [OSD 2015].

NIST SP 800-160

Systems Security Engineering, November 2016 [NIST 2018b].

DoDI 8510.01

Risk Management Framework (RMF) for DoD Information Technology (IT), March 12, 2014 [DoD 2016a].

Guidebooks

The following guidebooks are available to support PMs in concepts related to system engineering and software engineering.

DoD Program Manager Guidebook for Integrating the Cybersecurity Risk Management Framework (RMF) into the System Acquisition Lifecycle [DoD 2015c]

This guidebook emphasizes integrating cybersecurity activities into existing processes, including requirements, SSE, program protection planning, trusted systems and networks analysis, developmental and operational test and evaluation, financial management and cost estimating, and sustainment and disposal. It provides more of an outside-in view of the RMF for the PM on integrating cybersecurity activities into the system's acquisition lifecycle, while this guidebook provides more of an inside-out engineering perspective of what a PM must know about the engineering-in of software assurance activities. The guidebooks are compatible with each other and useful to PMs in carrying out their responsibilities in software assurance and cybersecurity risk management.

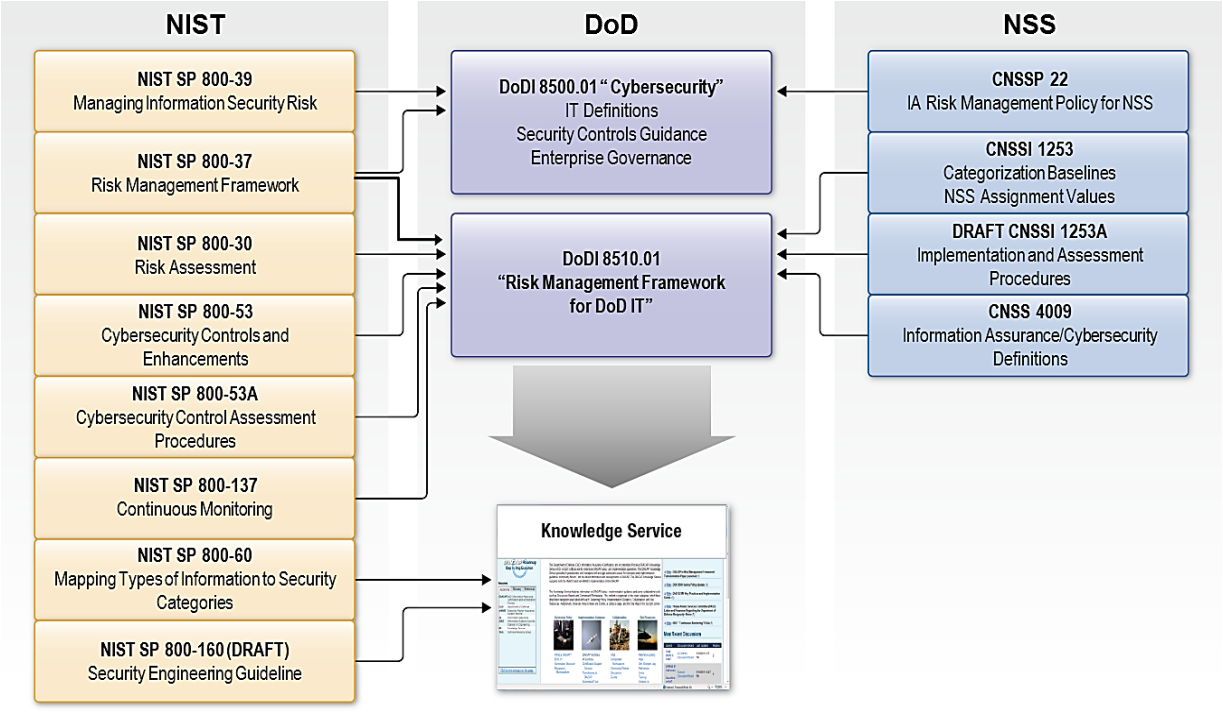
Engineering for Systems Assurance Guide [NDIA 2008]

This guide covers PM and system engineering assurance roles and responsibilities over the system engineering lifecycle. It includes the phases of the DoD Integrated Defense Acquisition, Technology, and Logistics Lifecycle Management System as discussed in DoD Directive 5000.01, DoDI 5000.02, the guidance in the DAG, and ISO/IEC 15288 Systems and Software Engineering – Systems Life Cycle processes.

Defense Acquisition Guidebook [DAU 2017]

The DAG is designed to complement formal acquisition policy as described in DoD Directive 5000.01 and DoDI 5000.02 by providing the acquisition workforce with discretionary best practices that should be tailored to the needs of each program. This guidebook is not a rule book or a checklist and does not require specific compliance with the business practice it describes. It is intended to inform thoughtful program planning and facilitate effective program management. In the area of software assurance, Chapters 3 and 9 are of interest. Chapter 3, Systems Engineering, describes standard system engineering processes and how they apply to the DoD acquisition system. Chapter 9, Program Protection, explains the actions needed to ensure effective program protection planning throughout the acquisition lifecycle.

Relationships of NIST, DoD, and NSS Publications and Policies



Derived with permission from a diagram originally developed by the Defense Acquisition University

Figure 11: Relevant DoD Publications and Policies

Appendix D: PM Checklist – Software Assurance by Phase

This PM Checklist was derived with permission from checklists originally developed by DASD(SE).

Tasks identified here need to be addressed early and assume previous items have been addressed and are in contract CDRLs. For example, when referring to checklist items in the Engineering and Manufacturing Development Phase, it is assumed that all checklist items in the Material Solution Analysis Phase and the Technology Maturation and Risk Reduction Phase have already been addressed.

Materiel Solution Analysis Phase

- Identify software assurance roles, responsibilities, and assurance needs for the program (e.g., staffing, tools, training); plan for software assurance training and resourcing.
- For the risk management process, develop understanding of how the deployed system may be attacked via software and use that understanding to scope criticality and threat analyses that are summarized in the PPP.
- Plan assessments and map tactical use threads, mission threads, system requirements, system interoperability, and functionality upgrades from the existing deployed system, and maintain the mapping as metadata through the last upgrade in sustainment.
- Identify system requirements that may map to software and software assurance requirements to facilitate tradeoffs and studies to optimize functional architecture and system design, and planning and resourcing to mitigate software vulnerabilities, risks, and lifecycle cost. For an integration-intensive system that relies substantially on non-development/COTS/GOTS items, trade-space analysis can provide important information to understand the feasibility of capability and mission requirements as well as assurance of the non-developmental software supply chain. Consider alternatives to refine the system concept of implementation and optimize for modularity and digital engineering; ensure that contract language for assurance reduces technical and programmatic risk. Support contracts should be part of this early solution analysis, to articulate/manage government technical data rights that later impact software assurance.
- Select secure design and coding standards for the program based on system functionality.
- Plan for and resource the use of automated tools that determine assurance or that detect vulnerabilities, defects, and weaknesses in requirements; allocation of requirements to functional architecture; functional architecture; allocation of functions to system design; system design; allocation of design modules to software design, coding, and unit testing; and integration testing.
- Identify JFAC software assurance service providers to assist with software assurance planning and services, and engage as necessary.

- Develop software assurance activities that are interconnected across the system lifecycle, and document them in the program software engineering planning document and the program Integrated Master Schedule (IMS).

Technology Maturation and Risk Reduction Phase

NOTE: Assumes items in Materiel Solution Analysis Phase have already been addressed.

- Incorporate software assurance requirements, tool use, metrics, and assurance thresholds into solicitations. Architectures, designs, and code developed for prototyping are frequently reused later in development.
- Assess system functional requirements and verification methods for inclusion of software assurance tools, methodologies, and remediation across the development lifecycle.
- Assess whether requirements for software assurance are correct and complete. Consider means of attack such as insiders and adversaries using malicious inserts, system characteristics, interoperability with other systems, mission threads, and other factors. Assure that mapping and traceability are maintained as metadata for use in all downstream assessments.
- Establish baseline architecture and review for weaknesses (e.g., use the CWE), susceptibility to attack (e.g., use the CAPEC), and likelihood of attack success considering each detected weakness; identify potential attack entry points and mission impacts. Consider which families of automated software assurance engineering tools are needed for vulnerability or weakness detection.
- Review architecture and design for adherence to secure design principles, and assess soundness of architectural decisions considering likely means of attack, programming language choices, development environments, frameworks, use of open source software, and other elements.
- Identify and mitigate technical risks through competitive prototyping while engineering in assurance. System prototypes may be physical or math models and simulations that emulate expected performance. High-risk concepts may require scaled models to reduce uncertainty too difficult to resolve purely by mathematical emulation. Software prototypes that reflect the results of key trade-off analyses should be demonstrated during the TMRR phase. These demonstrations will provide software performance data (e.g., latency, security architecture, integration of legacy services, graceful function degradation and re-initiation, and scalability) to inform decisions about maturity; further, EMD estimates (schedule and lifecycle cost) often depend on reuse of software components developed in TMRR; therefore, to prevent technical debt, software assurance considerations must be taken into account.
- Develop a comprehensive system-level architecture and design (address function integrity, assurance of the functional breakout, function interoperation, and separation of function) that cover the full scope of the system in order to maintain capabilities across multiple releases and provide the fundamental basis to fight through cyber attack. The program focused on a given software build, release, or increment may only produce artifacts for that limited scope; however, vulnerability assessments often interact, so apply vulnerability risks system-wide and across all builds, releases, increments, and interfaces to interoperating systems, and maintain through development and sustainment. A PDR, for example, must maintain this system-

level and longer-term, end-state perspective, as one of its functions is to provide an assessment of system maturity for the Milestone Decision Authority to assess prior to Milestone B.

- Involve non-developmental item vendors in system design in order to assure that functional integration addresses actual vendor product capabilities. In an integration-intensive environment, system models may be difficult to develop and fully exploit if many system components come from proprietary sources or commercial vendors with restrictions on data rights. Explore alternatives early, and consider model-based system engineering as the means to engineer-in assurance. Validating system performance and security assumptions may be difficult or even impossible. Proactive work with the vendor community to support model development informs downstream assessments, including during sustainment.
- Establish and manage entry and exit criteria for software assurance at each System Engineering Technical Review in order to properly focus the scope of the reviews and achieve usable assessment results and thresholds. Increasing knowledge and definition of elements of the integrated system design should include details of support and data rights.

Engineering and Manufacturing Development Phase

NOTE: Assumes items in Materiel Solution Analysis Phase and Technology Maturation and Risk Reduction Phase have already been addressed.

- Review architecture and design to assess against secure design principles, including system element and function isolation, least common mechanism, least privilege, fault isolation, graceful degradation, function re-initialization, input checking, and validation. These are the engineering bases that enable system resilience.
- Enforce secure coding practices through code inspection augmented by automated static and origin analysis tools and secure code standards for the languages used.
- Detect vulnerabilities, weaknesses, and defects in the software as close to the point of generation as possible; prioritize according to likelihood and consequence of use by an adversary, remediate, and regression test.
- Confirm that software assurance requirements, vulnerability remediation, and unresolved vulnerabilities are mapped to module test cases and to the final acceptance test cases. This provides a basis for assurance that will be used in downstream assessments and system changes in sustainment. Ensure that software-enabled mission functions and components receive rigorous automated software assurance tool assessment including static code analysis (SCA), origin analysis, and penetration and fuzz testing, including application of test coverage analyzers. Multiple SCA tools should be used.
- Ensure that Critical Design Review (CDR) software documentation represents the design, performance, and test requirements and includes development and software/systems integration facilities for coding and integrating the deliverable software and assurance operations on the integrated development environment (IDE). Software and systems used for developing computer software configuration items (e.g., simulations and emulations) should be assured whenever possible. Problem report metadata should include assurance factors such as CWE and CVE numbers wherever relevant so that data is usable for tracking, reporting, and assur-

ance assessments. Legacy problem report tracking information can be used to profile and predict which types of software functions may accrue what levels of problem reports. Assessments of patterns of problem reports or vulnerabilities among software components of the system can provide valuable information to support program resource and progress decisions.

- Address systems assurance (software, hardware, firmware, function, interoperability) up front and early rather than delaying it until later software builds. For a program using an incremental software development approach, technical debt may accrue within a given build and across multiple builds without a plan or resources to remediate code vulnerabilities as they are generated. Technical reviews, at both the system and build levels, should have a minimum viable requirements and architecture baseline that includes software assurance requirements and assured design architecture considerations, as well as ensuring fulfillment of a build-centric set of incremental review criteria and requirements that include assurance. This baseline should be retained for use through the last upgrade in sustainment. For build content that needs to evolve across builds, the PM and the systems engineer should ensure that system-level vulnerabilities, defects, and weaknesses are recorded and mitigated as soon as practical to ensure that any related development or risk reduction activities occur in a timely manner. Configuration management and associated change control/review boards can facilitate the recording and management of build information and mapped assurance metadata.
- Ensure that all defects and weaknesses are remediated or dispositioned before each developmental module is checked into configuration management.
- Install system components in a System Integration Lab and assess continuously for assurance considerations throughout EMD. Assurance considerations include version update and associated configurations for all COTS in the IDE, including assurance tools, operational assurance tools for the IDE, techniques using operational assurance tools to detect insider threats and malicious activity, and configuration control of the installed software and files. Details of the use of developmental system interfaces should be assessed and validated to ensure their scalability, suitability, and security for use. The emphasis in an integration-intensive system environment may be less on development and more on implementation and test. Progressive levels of integration, composition, and use should be obtained in order to evaluate ever higher levels of system performance, conduct automated penetration and fuzz testing, and ultimately encompass end-to-end testing based on user requirements and expectations. Assessment and test results should be maintained for downstream test activities and system changes. If the system is later breached, assurance metadata generated during EMD will be a basis to determine behavior, impacts, and remediations. “Glue” code and other scripted extensions to the system operational environment to enable capability should be handled in as rigorous a manner for assurance as any developed software, that is, kept under strong configuration management, scanned with multiple software assurance tools, and inspected; updates should be properly regression-tested and progressively integrated/tested.

Production and Deployment Phase

NOTE: Assumes items in Materiel Solution Analysis Phase, Technology Maturation and Risk Reduction Phase, and Engineering and Manufacturing Development Phase have already been addressed.

- Continue to enforce secure design and coding practices for all software changes, such as for installation modifications, through inspections and automated scans for vulnerabilities and weaknesses, and maintain assessment results.
- Conduct automated code vulnerability scans using SCA and origin assessment tools, reporting, and prioritization, and execute defect remediation consistent with program policy as system changes occur. Tool updates can detect additional vulnerabilities, and installations in deployment can change software characteristics or code.
- Conduct penetration testing using retained red-team or other automated test cases to detect any variations from expected system behavior.
- Maintain and enhance added automated regression tests for remediated vulnerabilities, and employ test coverage analyzers to ensure sufficient test coverage for remediation.
- Progressive deployment of an integration-intensive system provides infrastructure, services, and higher-level capabilities as each release is verified and validated. A rigorous release process includes configuration management and the use of regression test suites that include software assurance tools. The PM, systems engineer, software engineer, and systems security engineer should ensure user involvement in gaining understanding and approval of changes to design, functions, or operation that may result from vulnerability remediation.
- Synchronize and time block builds as much as possible to avoid forced upgrades or other problems at end-user sites. End-user sites that perform their own customization or tailoring of the system installation should ensure that changes are mapped from the standard configuration, recorded, and shared with the PMO or the integrator/developer so that problem reporting and resolution activities account for any operational and performance implications, and so vulnerability assessment data are updated. Any changes should be scanned at the deployment site with multiple software assurance tools to assure that no detectable vulnerabilities were inserted. This information will be necessary for assessments in detecting breaches and for remediation of breaches.

Operations and Support Phase

NOTE: Assumes items in Materiel Solution Analysis Phase, Technology Maturation and Risk Reduction Phase, Engineering and Manufacturing Development Phase, and Production and Deployment Phase have already been addressed.

- The software sustainment activity should take ownership of all assurance-related metadata and results in support of the PMO and system operation.
- Continue to enforce secure design and coding practices during sustainment through inspections and automated scans for vulnerabilities and weaknesses during sustainment system upgrades, revisions, Engineering Change Proposals, patches, and builds. System changes in sustainment can be as significant as new acquisitions.

- Continue to conduct automated code vulnerability scans, reporting, and prioritization, and execute defect remediation.
- Maintain and enhance automated regression tests for remediated vulnerabilities, and employ test coverage analyzers to assure sufficient test coverage.
- Continue to conduct penetration testing using retained red-team or other automated test cases to detect any variations from expected system behavior.
- Develop and use procedures to facilitate and ensure effective software configuration management and control. For example, require static and origin analysis scans for any changes to executable scripts or code, with all detected vulnerabilities remediated before the changes are approved. A defined block change or follow-on incremental development that delivers new or evolved capability, maintenance, security, safety, or urgent builds and upgrades to the field should be accomplished using this best practice. Procedures for updating and maintaining software on fielded systems often require individual user action and may require specific training. There are inherent security risks involved in installing or modifying software on fielded weapon systems used in tactical activities. These risks should be anticipated and remediated during the MSA phase. For example, the software would have been designed so that device update in tactical situations can be assured *in situ* to reduce or eliminate the opportunities for malicious insertion, corruption, or loss of software or data. Software updates to business and IT systems can also pose risks to operational availability through insider threats that should be anticipated and mitigated during the MSA phase. For example, scan glue code periodically and assess any unknown changes for malicious insertions, and scan all executable software or scripts whenever changes are applied. For any changes that impact system function, assess the design to maintain separation of function. PMs and systems and software engineers should implement procedures and tools to assure the supply chain in order to reduce risk and prevent malicious insertions. The supply chain includes sources for COTS, GOTS, and open source libraries.
- Maintain test cases previously developed for automated penetration and fuzz testing tools used during operational testing or red-team operations during system maintenance, and asynchronously conduct them to detect changes in system function, operation, or timing from the baseline. Changes can be the result of undetected and operational malicious inserts by insiders.
- Plan for system upgrades and updates timed to limit the proliferation of releases and therefore focus available maintenance, assurance, and support resources. In an integration-intensive environment, security upgrades, technical refreshes, and maintenance releases can proliferate, causing loss of situational awareness of assurance posture at end-user sites. Configuration management and regression testing should be used to ensure system integrity and to maintain detailed situational awareness.
- Use software assurance tools such as origin analysis and penetration testing to detect changes in operational configuration between the deployed site and the tested baseline.

Appendix E: PM Checklist – Software Assurance for Technical Reviews

This PM Checklist was derived with permission from checklists originally developed by DASD(SE).

Three system engineering technical reviews are particularly important to the development of all systems: the SRR, the PDR, and the CDR. When non-traditional or iterative/agile approaches to system acquisition are being used, the focus should be on achieving the intent of these technical milestone reviews in order to evaluate the system's progress and technical solutions related to software assurance. For additional information on agile acquisition and milestone reviews, see the SEI's publications on agile use in government [SEI 2011; Wrubel 2014].

The SRR ensures that the system under review is ready to proceed into initial system design. It ensures all system requirements and performance requirements derived from the ICD or draft CDD are defined and testable and that they are consistent with cost, schedule, risk, technology readiness, and other system constraints.

The PDR assesses the maturity of the preliminary design supported by the results of requirements tradeoffs, prototyping, and technology demonstrations during the TMRR phase. The PDR establishes the allocated baseline and confirms that the system under review is ready to proceed into detailed design (development of build-to drawings, software code-to documentation, and other fabrication documentation) with acceptable risk.

The CDR assesses design maturity, design build-to or code-to documentation, and remaining risks and establishes the initial product baseline.

System Requirements Review (SRR)

- Objectives
  - Recommendation to proceed into development with acceptable risk.
  - Understanding of top-level system requirements is adequate to support further requirements analysis and design activities.
  - Government and contractor mutually understand system requirements, including (1) the preferred materiel solution (including its support concept) from the MSA phase, (2) available technologies resulting from the prototyping efforts, and (3) maturity of interdependent systems.
- Software Assurance Success Criteria Checklist
  - ☐ Select automated tools for design, vulnerability scan/analysis, etc.
  - ☐ Establish facilities, tools, equipment, staff, and funding.
  - ☐ Confirm contractor System Engineering Management Plan includes software assurance roles and responsibilities.

- Determine security requirements for programming languages, architectures, development environment, and operational environment.
- Identify secure design principles to guide architecture and design decisions.
- Establish processes for ensuring adherence to secure design and coding standards.
- Develop plan for addressing software assurance in legacy code.
- Establish assurance requirements for software to deter, detect, react, and recover from faults and attacks.
- Perform initial software assurance reviews and inspections, and establish tracking processes for completion of assurance requirements.

Preliminary Design Review (PDR)

- Objectives
  - Recommendation that allocated baseline fully satisfies user requirements and developers are ready to begin detailed design with acceptable risk.
  - Allocated baseline is established such that design provides sufficient confidence to support 10 U.S. Code § 2366b certification.
  - Preliminary design and basic system architecture support capability need and affordability target achievement.
- Software Assurance Success Criteria Checklist
  - Determine that baseline fully satisfies user requirements.
  - Review architecture and design against secure design principles, including system element isolation, least common mechanism, least privilege, fault isolation, input checking, and validation.
  - Determine if initial Software Assurance Reviews and Inspections received from assurance testing activities capture requirements appropriately.
  - Confirm that software assurance requirements are mapped to module test cases and to the final acceptance test cases.
  - Establish automated regression testing procedures and tools as a core process.

Critical Design Review (CDR)

- Objectives
  - Recommendation is made to start fabricating, integrating, and testing test articles with acceptable risk.
  - Product design is stable. Initial product baseline is established.
  - Design is stable and performs as expected. Initial product baseline established by the system detailed design documentation confirms affordability/should-cost goals. Government control of Class I changes as appropriate.
- Software Assurance Success Criteria Checklist
  - Enforce secure coding practices through code inspection augmented by automated static analysis tools.

- Detect vulnerabilities, weaknesses, and defects in the software; prioritize; and remediate.
- Assure chain of custody from development through sustainment for any known vulnerabilities and weaknesses remaining and mitigations planned.
- Assure hash checking for delivered products.
- Establish processes for timely remediation of known vulnerabilities (e.g., CVEs) in fielded COTS components.
- Ensure planned software assurance testing provides variation in testing parameters, such as through application of test coverage analyzers.
- Ensure software functions and components receive rigorous test coverage.

Appendix F: Terminology

This appendix provides definitions of the key terms used in this guidebook and references for where other terms can be found. Most of these definitions come from the National Initiative for Cybersecurity Careers and Studies glossary, managed by the Department of Homeland Security (DHS); NIST Internal Report 7298: Glossary of Key Information Security Terms, managed by NIST; and other government documents.

The objectives for this glossary are to enable clearer communication and a common understanding of cybersecurity terms through the use of plain English and annotations in the definitions. Note that the cybersecurity field is rich in terminology, and the PM is well advised to seek clarification from several sources.

anti-tamper (AT) – System engineering activities (hardware and/or software techniques) designed into the system architecture to protect Critical Program Information against unwanted technology transfer (e.g., technology loss), countermeasure development, and capability/performance enhancement through system modification.

attack – An attempt to gain unauthorized access to system services, resources, or information, or an attempt to compromise system integrity. Any kind of malicious activity that attempts to collect, disrupt, deny, degrade, or destroy information system resources or the information itself.

attack surface – A set of ways in which an adversary can enter a system and potentially cause damage. The attack surface for a software environment is the sum of the different points (the "attack vectors") where an unauthorized user (the "attacker") can try to enter data to or extract data from an environment.

authentication – A security measure designed to establish the validity of a transmission, message, or originator, or a means of verifying an individual's authorization to receive specific categories of information.

availability – Ensuring timely and reliable access to and use of information. The property of being accessible and useable on demand by an authorized entity.

CIA triad of information security – A baseline standard for evaluating and implementing information security regardless of the underlying system or organization. The three core goals have distinct requirements and processes within each other.

- confidentiality: Ensures that data or an information system is accessed by only an authorized person. User IDs and passwords, access control lists, and policy-based security are some of the methods through which confidentiality is achieved.
- integrity: Assures that the data or information system can be trusted, is edited by only authorized persons, and remains in its original state when at rest. Data encryption and hashing algorithms are key processes in providing integrity.

- availability: Assures that data and information systems are available when required. Hardware maintenance, software patching/upgrading, and network optimization ensure availability.

Common Attack Pattern Enumeration and Classification (CAPEC) – A comprehensive dictionary and classification taxonomy of known attacks that can be used by analysts, developers, testers, and educators to advance community understanding and enhance defenses. The CAPEC effort provides a publicly available catalog of attack patterns along with a comprehensive schema and classification taxonomy. CAPEC is sponsored by US-CERT in the Office of Cybersecurity and Communications at DHS. CAPEC and the CAPEC logo are trademarks of The MITRE Corporation.

Common Vulnerabilities and Exposures (CVE) – A dictionary of common names (i.e., CVE Identifiers) for publicly known cybersecurity vulnerabilities. CVE's common identifiers make it easier to share data across separate network security databases and tools and provide a baseline for evaluating the coverage of an organization's security tools. The National Cybersecurity FFRDC, a federally funded research and development center operated by The MITRE Corporation, maintains the system with funding from the National Cybersecurity Division of DHS.

Common Weakness Enumeration (CWE) – A community-developed list of common software security weaknesses. CWE is a formal list of software weakness types created to (1) serve as a common language for describing software security weaknesses in architecture, design, or code; (2) serve as a standard measuring stick for software security tools targeting these weaknesses; and (3) provide a common baseline standard for weakness identification, mitigation, and prevention efforts. CWE is sponsored by US-CERT in the Office of Cybersecurity and Communications at DHS. CWE and the CWE logo are trademarks of The MITRE Corporation.

communications security (COMSEC) – Measures and controls taken to deny unauthorized persons information derived from telecommunications and to ensure the authenticity of such telecommunications. COMSEC includes cryptosecurity, transmission security, emission security, and physical security of classified material.

Critical Program Information (CPI) – Elements or components of a research, development, and acquisition program that, if compromised, could cause significant degradation in mission effectiveness; shorten the expected combat-effective life of the system; reduce technological advantage; significantly alter program direction; or enable an adversary to defeat, counter, copy, or reverse engineer the technology or capability [DoD 2015b].

cybersecurity – The prevention of damage to, protection of, and restoration of computers, electronic communications systems, electronic communications services, wire communication, and electronic communication, including information contained therein, to ensure their availability, integrity, authentication, confidentiality, and nonrepudiation [DoD 2014].

exploit – To take advantage of a weakness (or multiple weaknesses) to achieve a negative technical impact.

exposure – The condition of being unprotected, thereby allowing access to information or access to capabilities that an attacker can use to enter a system or network.

information assurance (IA) – Measures that protect and defend information and information systems by ensuring their availability, integrity, and confidentiality.

information system security manager (ISSM) – Individual responsible for the information assurance of a program, organization, system, or enclave.

information systems security (INFOSEC) – Protection of information systems against unauthorized access to or modification of information, whether in storage, processing, or transit, and against the denial of service to authorized users, including those measures necessary to detect, document, and counter such threats.

Initial Capabilities Document (ICD) – Documents the need for a materiel approach, or an approach that is a combination of materiel and non-materiel, to a specific capability gap.

Joint Federated Assurance Center (JFAC) – A federation of DoD organizations that have a shared interest in promoting software and hardware assurance in defense acquisition programs, systems, and supporting activities.

key performance indicator (KPI) – Quantifiable measure that can be used to evaluate success in achieving key system capabilities.

key performance parameter (KPP) – Key system capabilities that must be met in order for a system to meet its operational goals. The Capability Development Document and Capability Production Document identify the KPPs that contribute to the desired operational capability in a specific system.

operations security (OPSEC) – Systematic and proven process by which potential adversaries can be denied information about capabilities and intentions by identifying, controlling, and protecting generally unclassified evidence of the planning and execution of sensitive activities.

personal security (PERSEC) – Systematic and proven process for protecting personal information.

program protection – The integrating process for managing risks to advanced technology and mission system functionality from foreign collection, design vulnerability, supply-chain exploit/insertion, and battlefield loss throughout the acquisition lifecycle [DoD 2011].

Program Protection Plan (PPP) – The single source document used to coordinate and integrate all protection efforts. It is designed to deny access to Critical Program Information to anyone not authorized or who does not have a need to know and to prevent inadvertent disclosure of leading-edge technology to foreign interests.

project/program manager (PM) – The single individual responsible for a project or program who manages all daily aspects of the project or program.

Risk Management Framework (RMF) – A process that integrates security and risk management activities into the system development lifecycle.

Security Technical Implementation Guide (STIG) – A compendium of DoD policies, security regulations, and best practices for securing an IA or IA-enabled device (e.g., operating system, network, and application software). The Application Security STIG ensures that processes are in place to develop secure code.

software assurance – The level of confidence that software functions as intended and is free of vulnerabilities, either intentionally or unintentionally designed or inserted as part of the software, throughout the lifecycle [P.L. 112-239, Section 933(e)(2), 2013].

supervisory control and data acquisition (SCADA) – A generic name for a computerized system that is capable of gathering and processing data and applying operational controls over long distances. Typical uses include power transmission and distribution and pipeline systems. SCADA was designed for the unique communication challenges (delays, data integrity, etc.) posed by the various media that must be used, such as phone lines, microwaves, and satellites. Usually shared rather than dedicated.

supply-chain risk management (SCRM) – The implementation of strategies to manage both every-day and exceptional risks along the supply chain based on continuous risk assessment with the objective of reducing vulnerability and ensuring continuity.

system security engineering (SSE) – An element of system engineering that applies scientific and engineering principles to identify security vulnerabilities and minimize or contain risks associated with these vulnerabilities [DoD 2012].

system survivability KPP – A key performance parameter indicating that the system shall survive kinetic threats and electronic-warfare threats in a cyber-contested environment.

threat – Any circumstance or event with the potential to adversely impact agency operations (including mission, functions, image, or reputation), agency assets, or individuals through an information system via unauthorized access, destruction, disclosure, modification of information, and/or denial of service.

US-CERT – A partnership between the DHS and the public and private sectors, established to protect the nation's Internet infrastructure. US-CERT coordinates defense against and responses to cyber attacks across the nation.

vulnerability – Weakness in an information system, system security procedure, internal control, or implementation that could be exploited by a threat source [CNSS 2015].

weakness – A shortcoming or imperfection in software code, design, architecture, or deployment that, under proper conditions, could become a vulnerability or contribute to the introduction of vulnerabilities.

Appendix G: Acronyms and Abbreviations

This appendix provides acronyms and abbreviations used in this guidebook.

APT	advanced persistent threat
ASR	Acquisition Strategy Review
AT	anti-tamper
AT&L	Acquisition, Technology, and Logistics
ATO	Authority to Operate
ATT	Adversary Threat Tier
CALIT	Cybersecurity and Acquisition Lifecycle Integration Tool
CAPE	Office of Cost Assessment and Program Evaluation
CAPEC	Common Attack Pattern Enumeration and Classification
CDD	Capability Development Document
CDR	Critical Design Review
CDRL	Contract Data Requirements List
CI	Cyber Integrator
CIA	Confidentiality, Integrity, and Availability
CIO	chief information officer
COMSEC	communications security
CONOPS	concept of operations
COSMIC	Common Software Measurement International Consortium
COTS	commercial off-the-shelf
CPD	Capability Production Document
CPI	Critical Program Information
CSA	Cyber-Survivability Attribute
CVE	Common Vulnerabilities and Exposures

CWE	Common Weaknesses Enumeration
DAG	Defense Acquisition Guidebook
DASD	Deputy Assistant Secretary of Defense
DAU	Defense Acquisition University
DFARS	Defense Federal Acquisition Regulation Supplement
DHS	Department of Homeland Security
DIACAP	DoD Information Assurance Certification and Accreditation Process
DISA	Defense Information Systems Agency
DoD	Department of Defense
DoDI	Department of Defense Instruction
DoDIN	DoD Information Networks
DOT&E	Director of Operational Test & Evaluation
DT&E	developmental test and evaluation
EMD	Engineering and Manufacturing Development Phase
FAR	Federal Acquisition Regulation
GOTS	government off-the-shelf
IA	information assurance
IAVA	Information Assurance Vulnerability Alert
ICD	Initial Capabilities Document
IDE	integrated development environment
IEC	International Electrotechnical Commission
IMS	Integrated Master Schedule
INFOSEC	information systems security
IPS	Integrated Product Support
ISO	International Organization for Standardization
ISSM	information system security manager
ISSO	information system security officer

IT	information technology
IV&V	independent verification and validation
JCIDS	Joint Capabilities Integration and Development System
JFAC	Joint Federated Assurance Center
KPI	key performance indicator
KPP	key performance parameter
LCSP	Lifecycle Sustainment Plan
MDA	Milestone Decision Authority
MDAP	Major Defense Acquisition Program
MSA	Materiel Solution Analysis Phase
NDAA	National Defense Authorization Act
NDIA	National Defense Industrial Association
NIST	National Institute of Standards and Technology
NSA	National Security Agency
NVD	National Vulnerability Database
O&S	Operations and Sustainment Phase
OPSEC	operations security
OSD	Office of the Secretary of Defense
OT&E	Operational Test and Evaluation
OTA	Operational Test Agency
PD	Production and Deployment Phase
PDR	Preliminary Design Review
PEO	Program Executive Office
PERSEC	personnel security
PM	program manager
PMO	program management office
POA&M	Plan of Action and Milestones

PPP	Program Protection Plan
RDT&E	Research, Development, Test, and Evaluation
RFP	Request for Proposal
RMF	Risk Management Framework
SCA	static code analysis
SCADA	supervisory control and data acquisition
SCRM	supply-chain risk management
SDP	Software Development Plan
SE	systems engineering
SEP	System Engineering Plan
SFR	System Functional Review
SP	Special Publication
SRR	System Requirements Review
SSE	systems security engineering
SSWG	System Security Working Group
STIG	Security Technical Implementation Guide
T&E	test and evaluation
TMRR	Technology Maturation and Risk Reduction
TPM	technical performance measures
TSN	Trusted Systems and Networks
USD	Under Secretary of Defense
USD(AT&L)	Under Secretary of Defense for Acquisition, Technology, and Logistics

Appendix H: Tools, Techniques, and Countermeasures Throughout the Lifecycle

Table 4: Tools, Techniques, and Countermeasures, by Lifecycle Process

Tool/Technique	Stakeholder Requirements Definition	Requirements Analysis	Architectural Design	Implementation	Integration	Verification	Transition	Validation	Operation	Maintenance
Assurance Case		x	x							
ATAM			x							
Attack Modeling			x							
Automated Regression Test				x		x		x		x
Binary Weakness Analyzer				x	x	x		x		x
Binary/Bytecode Disassembler					x					x
Binary/Bytecode Simple Extractor				x						x
Bytecode Weakness Analyzer				x		x				x
Compare Binary/Bytecode to Application Permission Manifest									x	
Configuration Checker									x	
Coverage-Guided Fuzz Tester					x	x				x
Database Scanner									x	
Debugger				x	x	x				x
Development/Sustainment Version Control				x	x	x		x		x
Digital Forensics										x
Digital Signature Verification							x		x	
Execute and Compare with Application Manifest						x		x		
Fault Injection						x				
Firewall									x	
Focused Manual Spot Check				x		x				
Forced Path Execution						x				
Formal Methods/Correct by Construction				x						

Tool/Technique	Stakeholder Requirements Definition	Requirements Analysis	Architectural Design	Implementation	Integration	Verification	Transition	Validation	Operation	Maintenance
Framework-Based Fuzzer						x		x		
Fuzz Tester						x		x		
Generated Code Inspection				x						
Hardening Tools/Scripts									x	
Host Application Interface Scanner									x	
Host-Based Vulnerability Scanner									x	
IEEE 1028 Inspections				x						
Inter-application Flow Analyzer								x	x	
Intrusion Detection Systems/ Intrusion Prevention Systems									x	
Logging Systems									x	
Man-in-the-Middle Attack Tool								x		
Manual Code Review				x						x
Mission Thread Workshop	x									
Negative Testing				x		x		x		
Network Sniffer									x	
Network Vulnerability Scanner									x	
Obfuscated Code Detection				x						x
Origin Analyzer				x	x	x	x			
Penetration Test								x		
Permission Manifest Analyzer									x	
Probe-Based Attack with Tracked Flow										
Quality Attribute Workshop		x								
Rebuild and Compare				x					x	
Safer Languages			x	x						
Secure Library Selection			x	x						
Secured Operating System Overview				x					x	

Tool/Technique	Stakeholder Requirements Definition	Requirements Analysis	Architectural Design	Implementation	Integration	Verification	Transition	Validation	Operation	Maintenance
Security Information and Event Management									x	
Software Engineering Risk Analysis			x							
Source Code Knowledge Extractor				x						x
Source Code Quality Analyzer				x						
Source Code Weakness Analyzer				x						x
Test Coverage Analyzer										
Traditional Virus/Spyware Scanner				x					x	
Web Application Vulnerability Scanner									x	

Bibliography

[Alberts 2017]
Alberts, Christopher & Woody, Carol. *Prototype Software Assurance Framework (SAF): Introduction and Overview.* CMU/SEI-2017-TN-001. Software Engineering Institute, Carnegie Mellon University. 2017. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=496134

[Allen 2007]
Allen, Julia & Westby, Jody. *Governing for Enterprise Security (GES) Implementation Guide.* CMU/SEI-2007-TN-020. Software Engineering Institute, Carnegie Mellon University. 2007. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=8251

[ANSI/ISA 2008]
American National Standards Institute & Internet Security Alliance. The Financial Impact of Cyber Risk: 50 Questions Every CFO Should Ask. *Internet Security Alliance*. 2008. https://isalliance.org/publications/the-financial-impact-of-cyber-risk-50-questions-every-cfo-should-ask-2/

[Bartol 2008]
Bartol, Nadya et al. *Practical Measurement Framework for Software Assurance and Information Security*, Version 1.0. Practical Software & Systems Measurement (PSM). 2008. http://www.psmsc.com/Downloads/TechnologyPapers/SwA%20Measurement%2010-08-08.pdf

[Bianco 2007]
Bianco, Philip; Kotermanski, Rick; & Merson, Paulo. *Evaluating a Service-Oriented Architecture.* CMU/SEI-2007-TR-015. Software Engineering Institute, Carnegie Mellon University. 2007. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=8443

[Brown 2017]
Brown, Scott M. & Hurt, Thomas. Engineering Software Assurance into Weapons Systems During the DoD Acquisition Lifecycle. *Journal of Cyber Security and Information Systems.* Volume 5. Number 3. November 2017. https://www.csiac.org/journal-article/engineering-software-assurance-into-weapons-systems-during-the-dod-acquisition-life-cycle/

[Butler 2016]
Butler, Greg; Kendall, Kim; & Hunsinger, Stephani. *Cybersecurity in the Systems Engineering Process: A High Level SE Process Orientation*. Defense Acquisition University. June 2016. http://docplayer.net/37887240-Cybersecurity-in-the-systems-engineering-processes-a-high-level-se-process-orientation.html

[Caralli 2011]
Caralli, R. A.; Allen, J. H.; & White, D. W. *CERT Resilience Management Model: A Maturity Model for Managing Operational Resilience*. Addison-Wesley, 2011.

[Clark 2015]
Clark, Tim. Most Cyber Attacks Occur from This Common Vulnerability. *Forbes*. March 2015. https://www.forbes.com/sites/sap/2015/03/10/most-cyber-attacks-occur-from-this-common-vulnerability/#7f5ff12f7454

[CNSS 2014]
Committee on National Security Systems. *Security Categorization and Control Selection for National Security Systems*. Instruction 1253. CNSS. March 2014. https://www.cnss.gov/CNSS/openDoc.cfm?unl2+hDQ7g7taDD3FUFA9Q==

[CNSS 2015]
Committee on National Security Systems. *Committee on National Security Systems (CNSS) Glossary*. Instruction 4009. April 2015. https://www.cnss.gov/CNSS/openDoc.cfm?7enWQtHnsqMRcSGXsvHo8Q==

[Common Criteria 2017]
Common Criteria. *Common Criteria for Information Technology Security Evaluation*, Version 3.1, Revision 5. CCMB-2017-04-001. CCRA. April 2017. https://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R5.pdf

[DASD(SE) 2014]
Deputy Assistant Secretary of Defense for Systems Engineering and Department of Defense Chief Information Officer. *Suggested Language to Incorporate System Security Engineering for Trusted Systems and Networks into Department of Defense Requests for Proposals*. DoD. 2014. https://www.acq.osd.mil/se/docs/sse-language-for-tsn-in-dod-rfps.pdf

[DASD(SE) 2017]
Office of the Deputy Assistant Secretary of Defense for Systems Engineering. *Department of Defense Risk, Issue, and Opportunity Management Guide for Defense Acquisition Programs*. DoD. 2017. https://www.acq.osd.mil/se/docs/2017-RIO.pdf

[DAU 2017]
Defense Acquisition University. *Defense Acquisition Guidebook*. DAU. 2017. https://www.dau.mil/tools/dag

[Davis 2005]
Davis, Noopur. *Secure Software Development Life Cycle Processes: A Technology Scouting Report.* CMU/SEI-2005-TN-024. Software Engineering Institute, Carnegie Mellon University. 2005. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=7457

[DISA 2018]
Defense Information Systems Agency. *Data Rights FAQ*. DISA. 2018. https://disa.mil/About/Legal-and-Regulatory/DataRights-IP/DataRights

[DoD 2011]
Department of Defense. *Program Protection Plan Outline and Guidance*, Version 1.0. DoD. July 2011. https://www.acq.osd.mil/se/docs/PPP-Outline-and-Guidance-v1-July2011.pdf

[DoD 2012]
Department of Defense. *DoD Instruction 5200.44, Protection of Mission Critical Functions to Achieve Trusted Systems and Networks (TSN)*. DoD. November 2012. https://fas.org/irp/doddir/dod/i5200_44.pdf

[DoD 2014]
Department of Defense. *DoD Instruction 8500.01, Cybersecurity*. DoD. March 2014. https://fas.org/irp/doddir/dod/i8500_01.pdf

[DoD 2015a]
Department of Defense. *DoD Directive 5200.47E, Anti-Tamper (AT)*. DoD. September 2015. https://fas.org/irp/doddir/dod/d5200_47.pdf

[DoD 2015b]
Department of Defense. *DoD Instruction 5200.39, Critical Program Information (CPI) Identification and Protection within Research, Development, Test, and Evaluation (RDT&E)*. DoD. May 2015. https://fas.org/irp/doddir/dod/i5200_39.pdf

[DoD 2015c]
Department of Defense. *DoD Program Manager Guidebook for Integrating the Cybersecurity Risk Management Framework (RMF) into the System Acquisition Lifecycle*. DoD. October 2015. https://www.dau.mil/tools/t/DoD-Program-Manager-Guidebook-for-Integrating-the-Cybersecurity-Risk-Management-Framework-(RMF)-into-the-System-Acquisition-Lifecycle

[DoD 2015d]
Department of Defense. *Manual for the Operation of the Joint Capabilities Integration and Development System (JCIDS)*. DoD. February 2015.

[DoD 2016a]
Department of Defense. *DoD Instruction 8510.01, Risk Management Framework (RMF) for DoD Information Technology (IT)*. DoD. 2016. https://fas.org/irp/doddir/dod/i8510_01.pdf

[DoD 2016b]
Department of Defense. *Product Support Manager Guidebook*. DoD. 2016. https://www.dau.mil/guidebooks/Shared%20Documents%20HTML/PSM%20Guidebook.aspx

[DoD 2017]
Department of Defense. Enclosure 14: Cybersecurity in the Defense Acquisition System. Pages 171–187. *DoD Instruction 5000.02, Operation of the Defense Acquisition System.* DoD. February 2017. https://www.dau.mil/guidebooks/Shared%20Documents%20HTML/DoDI%205000.02.aspx#toc311

[DoD 2018a]
Department of Defense. *Cybersecurity Test and Evaluation Guidebook*, Version 2.0. DoD. April 2018. https://www.dau.mil/tools/t/DoD-Cybersecurity-Test-and-Evaluation-Guidebook

[DoD 2018b]
Department of Defense. *Department of Defense Directive 5205.02E: DoD Operations Security (OPSEC) Program* (Incorporating Change 1). DoD. May 2018. http://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodd/520502e.pdf?ver=2018-05-11-103259-050

[DoD 2018c]
Department of Defense. *Department of Defense Manual 5200.01: DoD Information Security Program* (Incorporating Change 1). DoD. 2018. https://www.hsdl.org/?view&did=810703

[DSB 2017]
Defense Science Board. *DSB Task Force on Cyber Supply Chain*. Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics. February 2017. https://www.acq.osd.mil/dsb/reports/2010s/1028953.pdf

[Ellison 2010]
Ellison, Robert; Goodenough, John; Weinstock, Charles; & Woody, Carol. *Evaluating and Mitigating Software Supply Chain Security Risks.* CMU/SEI-2010-TN-016. Software Engineering Institute, Carnegie Mellon University. 2010. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9337

[EOP 2011]
Executive Office of the President and the National Science and Technology Council. *Trustworthy Cyberspace: Strategic Plan for the Federal Cybersecurity Research and Development Program*. White House Office of Science & Technology Policy. December 2011. https://www.nitrd.gov/pubs/Fed_Cybersecurity_RD_Strategic_Plan_2011.pdf

[Fall 2003]
Fall, Kevin. A Delay-Tolerant Network Architecture for Challenged Internets. Pages 27–34. *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. ACM. 2003. https://dl.acm.org/citation.cfm?id=863960

[Fall 2008]
Fall, Kevin & Farrell, Stephen. DTN: An Architectural Retrospective. *IEEE Journal on Selected Areas in Communications*. Volume 26. Issue 5. June 2008. Pages 828–836. https://ieeexplore.ieee.org/document/4530739

[Fong 2016]
Fong, E. K. H.; Wheeler, D. A.; & Henninger, A. E. *State-of-the-Art Resources (SOAR) for Software Vulnerability Detection, Test, and Evaluation*. Institute for Defense Analysis. 2016. https://www.acq.osd.mil/se/docs/P-8005-SOAR-2016.pdf

[Giampapa 2012]
Giampapa, J. A. &. Hug-Glanzmann, G. *A Cyber-Physical and Agent-Based Defense to False Data Injection Attacks on a SCADA System*. Presented at the Great Lakes Symposium on Smart Grid and the New Energy Economy. September 2012. http://www.iitmicrogrid.net/event/great-lake2012/publication/PPTs/8-Smart%20Grid%20Cyber%20Security%20and%20Data%20Management/8-giampapaGreatLake2012_k2.pdf

[Glass 2011]
Glass, Kristin & Colbaugh, Richard. Web Analytics for Security Informatics. Pages 214–219. In *Proceedings of the 2011 European Intelligence and Security Informatics Conference.* September 2011. https://ieeexplore.ieee.org/document/6061237

[Goldsmith 2015]
Goldsmith, Rob & Mills, Steve. Cyber Integrator: A Concept Whose Time Has Come. *Defense AT&L Magazine*. March–April 2015. Pages 38–40.
http://www.dtic.mil/dtic/tr/fulltext/u2/a620544.pdf

[Goolsby 2013]
Goolsby, R. *On Cybersecurity, Crowdsourcing and Social Cyber-Attack* (Commons Lab Policy Memo Series, Vol. 1). Woodrow Wilson Center. 2013. http://www.wilsoncenter.org/sites/default/files/127219170-On-Cybersecurity-Crowdsourcing-Cyber-Attack-Commons-Lab-Policy-Memo-Series-Vol-1.pdf

[Hilburn 2013]
Hilburn, Thomas; Ardis, Mark; Johnson, Glenn; Kornecki, Andrew; & Mead, Nancy. *Software Assurance Competency Model.* CMU/SEI-2013-TN-004. Software Engineering Institute, Carnegie Mellon University. 2013. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=47953

[Howard 2005]
Howard, Michael; Pincus, Jon; & Wing, Jeannette M. Measuring Relative Attack Surfaces. *Computer Security in the 21st Century*. Edited by D. T. Lee, S. P. Shieh, & J. D. Tygar. Springer. Pages 109–137. 2005. https://link.springer.com/chapter/10.1007/0-387-24006-3_8

[Huang 2005]
Huang, L. & Boehm, B. Determining How Much Software Assurance Is Enough? A Value-Based Approach. Pages 172–181. In *Proceedings of the 2005 International Symposium on Empirical Software Engineering*. November 2005. https://ieeexplore.ieee.org/document/1541826

[IIC 2016]
Industrial Internet Consortium. *Industrial Internet of Things, Volume G4: Security Framework*. IIC. September 2016. https://www.iiconsortium.org/pdf/IIC_PUB_G4_V1.00_PB.pdf

[ISO/IEC 2015]
ISO/IEC JTC 1/SC 7. *ISO/IEC/IEEE Standard 15288 for Systems and Software Engineering – System Life Cycle Processes*. ISO. 2015. https://www.iso.org/standard/63711.html

[ISO/IEC 2017]
ISO/IEC JTC 1/SC 7. Section E4: Process View for Software Assurance. *ISO/IEC/IEEE Standard 12207 for Systems and Software Engineering – Software Life Cycle Processes*. ISO 2017. https://www.iso.org/standard/63712.html

[Jang 2012]
Jang, J.; Agrawal, A.; & Brumley, D. ReDeBug: Finding Unpatched Code Clones in Entire OS Distributions. Pages 48–62. In *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE Computer Society. May 2012. https://dl.acm.org/citation.cfm?id=2310707

[Jones 2011]
Jones, C. & Bonsignour, O. *The Economics of Software Quality*. Addison-Wesley Professional. 2011.

[Lewis 2012]
Lewis Grace; Novakouski, Marc; & Sánchez, Enrique. A Reference Architecture for Group-Context-Aware Mobile Applications. Pages 44–63. In *Proceedings of the 4th International Conference on Mobile Computing, Applications and Services*. October 2012. http://link.springer.com/chapter/10.1007/978-3-642-36632-1_3

[McGrew 2009]
McGrew, R. W. & Vaughn, R. B. Discovering Vulnerabilities in Control System Human–Machine Interface Software. *Journal of Systems and Software*. Volume 82. Issue 4. April 2009. Pages 583–589. http://www.sciencedirect.com/science/article/pii/S0164121209000077

[Mead 2005]
Mead, Nancy; Hough, Eric; & Stehney II, Ted. *Security Quality Requirements Engineering Technical Report.* CMU/SEI-2005-TR-009. Software Engineering Institute, Carnegie Mellon University. 2005. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=7657

[Mead 2010]
Mead, Nancy; Allen, Julia; Ardis, Mark; Hilburn, Thomas; Kornecki, Andrew; Linger, Richard; & McDonald, James. *Software Assurance Curriculum Project Volume I: Master of Software Assurance Reference Curriculum.* CMU/SEI-2010-TR-005. Software Engineering Institute, Carnegie Mellon University. 2010. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9415

[Mead 2011a]
Mead, Nancy; Allen, Julia; Ardis, Mark; Hilburn, Thomas; Kornecki, Andrew; & Linger, Richard. *Software Assurance Curriculum Project Volume III: Master of Software Assurance Course Syllabi.* CMU/SEI-2011-TR-013. Software Engineering Institute, Carnegie Mellon University. 2011. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9981

[Mead 2011b]
Mead, Nancy; Hawthorne, Elizabeth; & Ardis, Mark. *Software Assurance Curriculum Project Volume IV: Community College Education.* CMU/SEI-2011-TR-017. Software Engineering Institute, Carnegie Mellon University. 2011. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=10009

[Mead 2016]
Mead, Nancy, and Carol Woody. *Cyber Security Engineering: A Practical Approach for Systems and Software Assurance*. Addison-Wesley. 2016.

[MITRE 2018]
The MITRE Corporation. *Common Weakness Enumeration: A Community-Developed Dictionary of Software Weakness Types*. MITRE. 2018.

[NDIA 2008]
National Defense Industrial Association, Systems Assurance Committee. *Engineering for Systems Assurance Guide*. NDIA. 2008. https://www.acq.osd.mil/se/docs/sa-guidebook-v1-oct2008.pdf

[Nichols 2018]
Nichols, William; McHale, James; Sweeney, David; Snavely, William; & Volkmann, Aaron. *Composing Effective Software Security Assurance Workflows.* CMU/SEI-2018-TR-004. Software Engineering Institute, Carnegie Mellon University. 2018. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=528467

[NIST 2007]
National Institute of Standards and Technology. *Information Security Handbook: A Guide for Managers*. Special Publication 800-100. NIST. March 2007. https://csrc.nist.gov/publications/detail/sp/800-100/final

[NIST 2011]
National Institute of Standards and Technology. *Managing Information Security Risk: Organization, Mission, and Information System View*. Special Publication 800-39. NIST. March 2011. https://csrc.nist.gov/publications/detail/sp/800-39/final

[NIST 2015a]
National Institute of Standards and Technology. *Security and Privacy Controls for Federal Information Systems and Organizations*. Special Publication 800-53, Revision 4. NIST. 2015. https://csrc.nist.gov/publications/detail/sp/800-53/rev-4/final

[NIST 2015b]
National Institute of Standards and Technology. *Supply Chain Risk Management Practices for Federal Information Systems and Organizations*. Special Publication 800-161. NIST. 2015. https://csrc.nist.gov/publications/detail/sp/800-161/final

[NIST 2018a]
National Institute of Standards and Technology. *National Vulnerability Database*. NIST. 2018. https://nvd.nist.gov/

[NIST 2018b]
National Institute of Standards and Technology. *Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems*. Special Publication 800-160. NIST. 2018. https://csrc.nist.gov/publications/detail/sp/800-160/vol-1/final

[NRC 2010]
National Research Council. *Critical Code: Software Producibility for Defense*. National Academies Press. 2010. http://www.nap.edu/catalog.php?record_id=12979

[OSD 2015]
Office of the Under Secretary of Defense. *Deputy Secretary of Defense Policy Memorandum (PM) 15-001–Joint Federated Assurance Center (JFAC) Charter*. OSD. February 2015.

[OUSD(AT&L) 2017]
Office of the Under Secretary of Defense for Acquisition, Technology and Logistics. *Report of the Defense Science Board Task Force on Cyber Supply Chain*. DoD. February 2017. https://www.acq.osd.mil/dsb/reports/2010s/1028953.pdf

[OWASP 2016]
Open Web Application Security Project. *OWASP Secure Software Contract Annex*. OWASP. March 2016. https://www.owasp.org/index.php/OWASP_Secure_Software_Contract_Annex

[P.L. 111-383 2011]
Public Law 111-383. National Defense Authorization Act (NDAA) for Fiscal Year 2011. Section 932, Strategy on Computer Software Assurance. January 2011. https://www.gpo.gov/fdsys/pkg/PLAW-111publ383/content-detail.html

[P.L. 112-239 2013]
Public Law 112-239. National Defense Authorization Act (NDAA) for Fiscal Year 2013. Section 933, Improvements in Assurance of Computer Software Procured by the Department of Defense. January 2013. https://www.gpo.gov/fdsys/pkg/PLAW-112publ239/content-detail.html

[P.L. 113-66 2013]
Public Law 113-66. National Defense Authorization Act (NDAA) for Fiscal Year 2014. Section 937, Joint Federated Centers for Trusted Defense Systems for the Department of Defense. December 2013. https://www.gpo.gov/fdsys/pkg/PLAW-113publ66/content-detail.html

[P.L. 115-91 2017]
Public Law 115-91. National Defense Authorization Act (NDAA) for Fiscal Year 2018. Section 871, Noncommercial Computer Software Acquisition Considerations. December 2017. https://www.gpo.gov/fdsys/pkg/PLAW-115publ91/content-detail.html

[PMI 2013]
Project Management Institute. *Software Extension to the PMBOK*. PMI. 2013.

[PMI 2017]
Project Management Institute. *Project Management Body of Knowledge*. PMI. 2017.

[Scherlis 2012]
Scherlis, William. Software Producibility for Defense [blog post]. *SEI Insights*. June 2012. https://insights.sei.cmu.edu/sei_blog/2012/06/software-producibility-for-defense.html

[Schneier 1999]
Schneier, Bruce. Attack Trees. *Dr. Dobbs Journal*. December 1999. https://www.schneier.com/academic/archives/1999/12/attack_trees.html

[Seacord 2012]
Seacord, Robert; Dormann, Will; McCurley, James; Miller, Philip; Stoddard, Robert; Svoboda, David; & Welch, Jefferson. *Source Code Analysis Laboratory (SCALe).* CMU/SEI-2012-TN-013. Software Engineering Institute, Carnegie Mellon University. 2012. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=10091

[SEI 2017]
Software Engineering Institute. *Agile Acquisition and Milestone Reviews*. Acquisition & Management Concerns for Agile Use in Government Series. Software Engineering Institute, Carnegie Mellon University. 2017. https://resources.sei.cmu.edu/library/asset-view.cfm?assetID=506342

[Serbu 2012]
Serbu, Jared. Pentagon Struggles to Secure a Supply Chain It No Longer Dominates. *Federal News Network*. September 2012. https://federalnewsnetwork.com/federal-drive/2012/09/pentagon-struggles-to-secure-a-supply-chain-it-no-longer-dominates/

[Shevchenko 2018]
Shevchenko, Nataliya; Chick, Timothy A.; O'Riordan, Paige; Scanlon, Tom; & Woody, Carol. *Threat Modeling: A Summary of Available Methods*. Software Engineering Institute, Carnegie Mellon University. August 2018. https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=524448

[Shin 2013]
Shin, Yonghee & Williams, Laurie. Can Traditional Fault Prediction Models Be Used for Vulnerability Prediction? *Empirical Software Engineering*. Volume 18. Issue 1. February 2013. Pages 25–59. http://www.springerlink.com/openurl.asp?genre=article&id=doi:10.1007/s10664-011-9190-8

[Shoemaker 2013]
Shoemaker, Dan & Mead, Nancy. *Software Assurance Measurement – State of the Practice.* CMU/SEI-2013-TN-019. Software Engineering Institute, Carnegie Mellon University. 2013. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=72885

[SMC 2018]
Office of the Staff Judge Advocate, Space and Missile Systems Center. *Acquiring and Enforcing the Government's Rights in Technical Data and Computer Software Under Department of Defense Contracts: A Practical Handbook for Acquisition Professionals*, 9th ed. 2018. https://www.dau.mil/tools/t/TDR-GB

[USD(AT&L) 2011]
Undersecretary of Defense, Acquisition, Technology, and Logistics. *Memorandum: Document Streamlining – Program Protection Plan (PPP)*. DoD. July 2011. https://www.acq.osd.mil/se/docs/PDUSD-ATLMemo-Expected-Bus-Practice-PPP-18Jul11.pdf

[Wilson 2017]
Wilson, Roy. Software Assurance Course (CLE 081). Defense Acquisition University. December 2017.

[Woody 2014]
Woody, Carol; Ellison, Robert; & Nichols, William. *Predicting Software Assurance Using Quality and Reliability Measures.* CMU/SEI-2014-TN-026. Software Engineering Institute, Carnegie Mellon University. 2014. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=428589

[Wrubel 2014]
Wrubel, Eileen; Miller, Suzanne; Lapham, Mary Ann; & Chick, Timothy. *Agile Software Teams: How They Engage with Systems Engineering on DoD Acquisition Programs.* CMU/SEI-2014-TN-013. Software Engineering Institute, Carnegie Mellon University. 2014. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=295943

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	December 2018	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Program Manager's Guidebook for Software Assurance	FA8702-15-D-0002

6. AUTHOR(S)

Ken Nidiffer, Carol Woody, Tim Chick

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2018-SR-025

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
SEI Administrative Agent AFLCMC/AZS Enterprise Acquisition Division 5 Eglin Street Hanscom AFB, MA 01731-2100	n/a

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

The *Program Manager's Guidebook for Software Assurance* supports project managers who must integrate software assurance engineering activities into the acquisition lifecycle. The goal of the guidebook is to help the program manager (PM) understand and address the software assurance responsibilities critical in defending software-intensive systems. It presents actions a PM must take to ensure that software assurance is effectively addressed. These actions require an understanding of program mission threads, threat awareness, and the roles and responsibilities of members of the program office team. The guidebook objectives are aligned with (1) Enclosure 14 of Department of Defense (DoD) Instruction 5000.02, which provides policies and principles for cybersecurity in defense acquisition systems; (2) the Defense Acquisition University's Software Assurance Course (CLE 081); (3) the DoD Integrated Defense Acquisition, Technology, and Logistics Lifecycle; and (4) the Deputy Assistant Secretary of Defense (Systems Engineering) Software Assurance Concept of Operations.

14. SUBJECT TERMS	15. NUMBER OF PAGES
acquisition, project management, software assurance, software engineering	98

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. Z39-18
298-102