ENGINEERING FOR SYSTEM ASSURANCE

Version 1.0

National Defense Industrial Association
System Assurance Committee

ENGINEERING FOR SYSTEM ASSURANCE

Version 1.0

National Defense Industrial Association
System Assurance Committee

National Defense Industrial Association
Arlington, Virginia

National Defense Industrial Association
System Assurance Committee Chairs
Kristen J. Baldwin, Acting Director, Systems and Software Engineering, Office of the Deputy Under Secretary of Defense for Acquisition and Technology
Mitchell Komaroff, Director, Globalization Task Force, Office of the Assistant Secretary of Defense for Networks and Information Integration
Paul Croll, CSC

Engineering for System Assurance

Citation should appear as follows:
National Defense Industrial Association (NDIA) System Assurance Committee. 2008. *Engineering for System Assurance*. Arlington, VA: NDIA.

National Defense Industrial Association
2111 Wilson Boulevard, Suite 400
Arlington, VA 22201

To submit questions or corrections, please contact
Office of the Deputy Under Secretary of Defense
for Acquisition and Technology
Systems and Software Engineering
Software Engineering and System Assurance
3090 Defense Pentagon, Washington, DC 20301

CONTENTS

FIGURES

TABLES

Contributors

Sponsors

Mitchell Komaroff, Office of the Assistant Secretary of Defense for Networks and Information Integration (OASD/NII)

Kristen J. Baldwin and E. Kenneth Hong Fong, Office of the Under Secretary of Defense for Acquisition, Technology and Logistics (OUSD(AT&L))

Paul Croll, CSC

Principal Authors *(in alphabetical order)*

Marie Stanley Collins, MITRE (OUSD(AT&L))

Terry Devine, MITRE (OUSD(AT&L))

Murray Donaldson, Decisive Analytics (OUSD(AT&L))

John Miller, MITRE (OASD/NII)

Rama Moorthy, Institute for Defense Analyses (OASD/NII)

Paul Popick, Aerospace Corporation

David Wheeler, Institute for Defense Analyses (OASD/NII)

Significant Contributors *(in alphabetical order)*

Reg Bartholomew, Rockwell Collins

Brian Cohen, Institute for Defense Analyses (OUSD(AT&L))

Archibald McKinlay, NOSSA, N32, Software Systems Safety Engineering

Samuel Redwine, James Madison University

Contributing Organizations

The following organizations provided comments on the draft guide. All comments were adjudicated and incorporated to the extent possible for this release. Any comments not fully reflected in this version will be addressed in future revisions.

Commenting Organization *(in alphabetical order by organization name provided)*	***Name*** *(if provided)*
Aerospace Corporation/INCOSE	David Beshore
AIA	Rusty Rentsch
Air Force Center for Systems Engineering	Mike Ucchino
Air Force Center for Systems Engineering	Randy Bullard
Army	Robert Schwenk
AT&L/SSE Contractor Support	Mike Zsak
BAE Systems	Barbara Jackson
BAE Systems	Dan Wiener
BAE Systems	J. LaBrosse
BAE Systems	Richard Bray
Boeing	Leslie V. Unruh
Boeing/INCOSE	Sharon Schmitt
Booz Allen Hamilton	Bob Trapp

continued

Commenting Organization *(in alphabetical order by organization name provided)*	***Name*** *(if provided)*
Booz Allen Hamilton	Larry Feldman
Booz Allen Hamilton	Nima Khamooshi
Booz Allen Hamilton	David Kleiner
Booz Allen Hamilton (OASD(NII))	William H. Edwards
CIO/G-6 AONS	G. Stiles
Cloakware	James Stibbards
Defense Contract Management Agency	L. Cianciolo
Defense Contract Management Agency	P. Young
Defense Logistics Agency	Not Named
Defense Procurement and Acquisition Policy	Not named
Dell Financial Services/INCOSE	Kwabby Gyasi
DoD GTF / MITRE	John Peshinski
DoD GTF / MITRE	Not named
DOT&E	Bill McCarthy
EMC	Dan Reddy
ESC	Steve Duncan
Harris/IA	Ronda Henning
IA/GCSD	Karen DiPaula
IA/GCSD	William Wall
IBM	Not named
Institute for Defense Analyses	Vashisht Sharma
Institute for Defense Analyses	Cliff Lau
ITAA	Community
ITAA	Trey Hodgkins
Jacobs Technology	Tom Nelson
JCS J5/DDGE/CSD	CDR James Imanian
JCS J6/J6I	Lt. Col. Jonathan Sutherland
Lockheed Martin	Michele Hanna
MCSC/ PM AAVS	Not named
MCSC/ PM Ammo	Moseley
Microsoft	Not Named
Microsoft	Not Named
Microsoft	Scott Charney
MITRE	Bob Natale
MITRE	Rich Pietravalle
NAVAIR	Ginn, Robert C CIV
NAVAIR - AIR-4.1.1.7	John Funk
NAVAIR - AIR-4.1F	Ken Goff
NAVAIR 7.2.6	Rozenbroek
Naval Surface Warfare Center Dahlgren	Chris Pettit

continued

Commenting Organization *(in alphabetical order by organization name provided)*	***Name*** *(if provided)*
Naval Surface Warfare Center Dahlgren	Adam Simonoff
Navy	Janet Gill
Navy Anti-Tamper	Donald R. Traeger
NIST	Marianne Swanson
Northrop Grumman	Christopher Meawad
NSA	Angela Weiland
OASD(NII)	Thomas Hickok
ODASD(I&IA)DIAP	Art King
ODUSD(I&E)	Patricia Huheey
OPNAV CIO	Not Named
OUSD(AT&L)SSE	Robert Skalamera
OUSD(AT&L)SSE/AS (DAC)	Chuck Johnson
OUSD(AT&L)SSE/AS (SAIC)	Wayne Young
OUSD(AT&L/SSE/DT&E (SAIC)	Bo Tye
OUSD(AT&L)SSE/ED	Not named
OUSD(AT&L)SSE/SSA (HPTi)	Christine Hines
OUSD(AT&L)SSE/SSA (HPTi)	Chris Powell
PEO Aviation	Terry Carlson
PEO Aviation	T. Grayson
PEO-AVN	Angela Hughes
PEO-AVN	Stephen Laws
PEO C4I, PMW 160	David Crotty
PEO IWS 7D	Will Kenney
PM WIN-T	David Mason
Raytheon/INCOSE	Elizabeth Wilson
Raytheon Systems	Edwin Lee
Rockwell Collins	Reg Bartholomew
SAF/AQLS	Lt. Col Edward Conant
SAF/AQLL	Kent Miller
SAFECode	Paul Kurtz
SE Validation Limited/INCOSE	Colin Brain
SE Validation/INCOSE	S M Crowe
SE Validation Limited	Richard Maguire
SEI	Carol Woody
SPAWAR 5.1.8	Mike Davis
SSC-CH	Karl Baker
SSC-SD	Mark Mason
Symantec Corporation	Wesley H. Higaki
Utility Helicopters Project Management Office (QuantiTech, Inc.)	LuAnn Lusk

This page intentionally left blank.

Executive Summary

For decades, industry and defense organizations have tried to build affordable, secure, and trustworthy systems. Despite significant strides toward this goal, there is ample evidence that adversaries retain their ability to compromise systems. Adversaries have a broad arsenal that includes:

- Insider attacks
- Social engineering attacks
- Physical attacks
- Attacks on the global supply chain
- Cyber attacks

Their attacks are often very hard to prevent or even detect. Consequently, there is growing awareness that systems must be designed, built, and operated with the expectation that system elements will have known and unknown vulnerabilities. Systems must continue to meet (possibly degraded) functionality, performance, and security goals despite these vulnerabilities. In addition, vulnerabilities should be addressed through a more comprehensive life cycle approach, delivering system assurance, reducing these vulnerabilities through a set of technologies and processes.

System assurance is *the justified confidence that the system functions as intended and is free of exploitable vulnerabilities, either intentionally or unintentionally designed or inserted as part of the system at any time during the life cycle*. This ideal of no exploitable vulnerabilities is usually unachievable in practice, so programs must perform risk management to reduce the probability and impact of vulnerabilities to acceptable levels.

This confidence is achieved by system assurance activities, which include a planned, systematic set of multi-disciplinary activities to achieve the acceptable measures of system assurance and manage the risk of exploitable vulnerabilities. The assurance case is the enabling mechanism to show that the system will meet its prioritized requirements, and that it will operate as intended in the operational environment, minimizing the risk of being exploited through weaknesses and vulnerabilities. It is a means to identify all the assurance claims, and from those claims trace through to their supporting arguments, and from those arguments to the supporting evidence. The assurance case is a critical mechanism for supporting the risk management process.

Today, the risk management process has often not considered assurance issues in an integrated way, resulting in project stakeholders unknowingly accepting assurance risks that can have severe financial, legal, and national security implications. Addressing assurance issues late in the process, or in a non-integrated way, may result in the system not being used as intended, or in excessive costs or delays in system certification or accreditation. Risk management must consider the broader consequences of a failure (e.g., to the mission, people's lives, property, business services, or reputation), not just the failure of the system to operate as intended. In many cases, layered defenses (e.g., defense-in-depth or engineering-in-depth) may be necessary to provide acceptable risk mitigation.

This guidebook provides process and technology guidance to increase the level of system assurance. This guidebook is intended primarily to aid program managers (PMs) and systems engineers (SEs) who are seeking guidance on how to incorporate assurance measures into their system life cycles. Assurance for security must be integrated into the systems engineering

activities to be cost-effective, timely, and consistent. In systems engineering, the activities for developing and maintaining the assurance case enable rational decision making, so that only the actions necessary to provide adequate justification (arguments and evidence) are performed. This guidebook is a synthesis of knowledge gained from existing practices, recommendations, policies, and mandates. System assurance activities are executed throughout the system life cycle.

This guidebook is organized based on ISO/IEC 15288:2008, Systems and software engineering – System life cycle processes. While there are other life cycle frameworks, this ISO/IEC standard combines a suitably encompassing nature while also providing sufficient specifics to drive system assurance. Those who use other life cycle frameworks should find it easy to map their frameworks to the ISO framework. Unless otherwise noted, cross-references refer to sections of this guidebook.

This guidebook also provides an assurance guidance section for use by the U.S. Department of Defense (DoD) and DoD contractors and subcontractors (see Section 4). This information is organized according to phases of the DoD Integrated Defense Acquisition, Technology and Logistics Life Cycle Management Framework discussed in DoD Directive 5000.1, DoD Instruction 5000.2, and the Defense Acquisition Guidebook (DAG) system life cycle. Future editions of this guidebook may add additional domain-specific assurance guidance.

The guidebook is ready for immediate use by PMs and SEs and is expected to expand and mature over time.

1 INTRODUCTION

1.1 Background

Despite significant strides in developing secure systems, adversaries continue to compromise systems. Adversaries have a broad arsenal of attacks that they can use during the development and operation of systems, including:

- Insider attacks
- Social engineering attacks
- Physical attacks
- Attacks on the global supply chain
- Cyber attacks

Developing effectively secure systems requires a combination of approaches to ensure mission success, including minimizing vulnerabilities and managing the remaining vulnerabilities. These approaches should be driven by threat analysis and oriented to mission success—in other words, they should address the prioritized threats that will most impact mission success.

1.2 Definition of System Assurance

For the purposes of this guidebook, system assurance (SA) is *the justified confidence that the system functions as intended and is free of exploitable vulnerabilities, either intentionally or unintentionally designed or inserted as part of the system at any time during the life cycle*. This ideal of no exploitable vulnerabilities is usually unachievable in practice, so programs must perform risk management to reduce the probability and impact of vulnerabilities to acceptable levels.

This confidence is achieved by system assurance activities, which include a planned, systematic set of multidisciplinary activities to achieve the acceptable measures of system assurance and manage the risk of exploitable vulnerabilities.

1.3 Purpose

The purpose of this guidebook is to provide guidance in how to build assurance into a system throughout its life cycle. This guidebook identifies and discusses systems engineering activities, processes, tools, and considerations to address system assurance. To be efficient, assurance issues must be addressed as early as possible; system assurance is often much more expensive to add to systems later in the life cycle. Assurance efforts should be commensurate with mission needs and threats (both identified and predictable). This guidebook is intended to be a living document. New techniques and tools will be addressed as they emerge.

1.4 Scope

This guidebook identifies processes, methods, techniques, activities, and tools for system assurance, using a systems engineering approach. As defined in ISO 15288, a system is a combination of interacting elements organized to achieve one or more stated purposes (missions). A system may be a system of systems or a family of systems (SoS/FoS).

This guidebook focuses on the electronic hardware (HW), firmware, and software (SW) elements that make up the system, including information storage, electronic sensor elements, information processors, and computer and communication elements. It focuses on the identification of implementable techniques, activities, and tools that can be applied to system assurance at the system and system element level. Where applicable, best practices and approved guidelines are referenced.

This guidebook discusses system assurance by specifically addressing the assurance of security properties throughout the system life cycle. These properties include confidentiality, integrity, availability, authentication, accountability (including non-repudiation), and auditability. It does not address assurance for quality, safety, or dependability, as there are existing documents that address those issues. However, an intelligent adversary may be able to subvert a system's functionality, quality, safety, or dependability if there is inadequate assurance of security properties.

This guidebook focuses on assurance of the entire system, not merely of specific system elements. Systems are normally composed from many elements—some commercial and some custom—with many different levels of assurance. Some elements may be "high assurance," meaning that compelling evidence is provided that the element delivers its services in a manner that satisfies certain critical properties (including compelling evidence that there are no software defects that would interfere with those properties). Developing software for high-assurance elements often relies on formal methods, which are rigorous, mathematically based techniques and tools for specifying, designing, and verifying hardware and software systems (Butler 2006), as well as extensive testing. Some elements may be "medium assurance," meaning that the element has been designed to meet its critical properties, and that significant effort has been expended to detect and address potential failures to meet critical properties (but not to the level of a high-assurance element). The assurance of an entire system depends on some (or all) of the system elements, but assuring specific elements is insufficient—the system must be considered as a whole. System developers may leverage specific high-assurance elements (so others need less assurance), design the system (e.g., by limiting privileges) so that weaknesses in one element will not harm system assurance, or use compensating processes. A *systems view* is vital for achieving system assurance.

1.5 Document Overview

Following this Introduction, Section 2 presents key concepts of system assurance that will appear throughout the guide. The section provides a foundation of the components that make up an assurance case and its value to the system development life cycle.

Section 3 provides general SA guidance that applies to all systems. Whenever possible, this section follows the structure of the ISO/IEC 15288:2008, Systems and Software Engineering–System Life Cycle Processes. Section 3.1, Agreement Processes, focuses on acquisition and supply. Section 3.2 discusses Organizational Project-Enabling Processes. Section 3.3, Project Processes, includes configuration management and risk management processes. Section 3.4, Technical Processes, discusses the recommended system assurance engineering activities from stakeholder requirements through disposal. Within each process, there is a dedicated subsection

called “building the assurance case,” which describes key activities that are required for building a system case.

Section 4 provides an assurance guidance section for use by the U.S. Department of Defense (DoD) and DoD contractors and subcontractors. This information is organized according to phases of the DoD Integrated Defense Acquisition, Technology and Logistics Life Cycle Management Framework discussed in DoD Directive (DoDD) 5000.1, The Defense Acquisition System; DoD Instruction (DoDI) 5000.2, Operation of the Defense Acquisition System; and the Defense Acquisition Guidebook (DAG) system life cycle. Future editions of this guidebook may add additional domain-specific assurance guidance.

Appendix A, Relationship of Safety, Security Efforts, Documents, Standards, includes a chart outlining safety and security references. Appendix B, Correspondence with Existing Documentation, Policies, and Standards, discusses related standards, policies, and laws. Appendix C maps DoD controls to guidebook information.

The guide also includes a glossary, references, acronym list, and index. Unless otherwise noted, cross-references refer to sections of this guide.

2 KEY SYSTEM ASSURANCE CONCEPTS

2.1 Key References

The major sources for system assurance guidance include the ISO/IEC Standard 15288 (2002), Systems Engineering System Life Cycle Processes; Goertzel et al. (2006); Redwine (2006), and the Common Weakness Enumeration (CWE).

2.2 Assurance Case

The purpose of an assurance case is to provide convincing justification to stakeholders that critical system assurance requirements are met in the system's expected environment(s). Should a set of system assurance claims about the system be expressed, these claims need to be incorporated into the system requirements. An assurance case is the set of claims of critical system assurance properties, arguments that justify the claims (including assumptions and context), and evidence supporting the arguments. The development of the assurance case results in system assurance requirements that are then flowed to the system architecture and the product baseline. The assurance case is generated by the systems engineering technical activities applied to the assurance requirements, and provides evidence of the growing technical maturity of the integration of the system assurance requirements for use within event-driven technical management.

In systems engineering, the activities for developing and maintaining the assurance case enable rational decision making, so that only the actions necessary to provide adequate justification (arguments and evidence) are performed. Assurance case planning identifies and justifies what approach will be taken (e.g., programming language selection, trusted source selection) and what evidence must be collected to justifiably achieve the required assurance. This planning includes cost and technical trade-offs, and integration into the risk mitigation process, the work breakdown structure, and the program/project schedule. Developing an assurance case clarifies the interaction between functionality, cost, schedule, security, safety, dependability, and other "-ilities", so that appropriate trade-offs can be made through risk management.

In systems engineering, assurance case development and maintenance should be executed as part of the stakeholder requirements definition, requirements analysis, architectural design, and risk management processes. The assurance case supports efficient system development while reducing overall risk, by applying various existing expertise/disciplines/processes to address the prioritized assurance requirements.

The assurance case need not be a separate document; it may be distributed among or embedded in existing documents. Even if there is an "assurance case" document, it would typically contain many references to other documents. Regardless of how the assurance case documentation is implemented, there must be a way to identify all the assurance claims, and from those claims trace through to their supporting arguments, and from those arguments to the supporting evidence. For example, an organization might maintain a list of system requirements, tagging specific ones as assurance claims. Each claim might have a hyperlink to the argument that justifies why the claim will be met (e.g., a risk-mitigation plan) as well as being flowed to the architecture and implementation baselines. That argument might in turn contain hyperlinks to evidence (e.g., results demonstrating that planned actions were successfully executed). The

assurance case shows that the system will meet the prioritized requirements, and that it will operate as intended in the operational environment, minimizing the risk of being exploited through weaknesses and vulnerabilities. During initial project planning, the Work Breakdown Structure (WBS) should include an activity to provide this traceability (see Section 3.3.1, Project Planning).

As a minimum:

1. Assurance case claims, arguments, and evidence must be relevant for the system and its operating environment(s)
2. Claims are justified by their arguments
3. Arguments are supported by their evidence
4. The assurance case must be developed iteratively, be sustainable, and be maintained throughout the system life cycle as a living document. This implies that the assurance should be developed in a way that makes it easy to change, that tools should be used to maintain it, and that it should be divided into smaller modules so changes can be localized.
5. The assurance case must be delivered as part of the system, to be maintained during system sustainment (Note: Some systems fail to do this, potentially invalidating the assurance case)

Many industry documents include in the term “assurance” the qualities of safety, security, and dependability. There are existing standards, best practices, and processes that support safety and dependability assurance. This guidebook assumes those processes pre-exist, but they may not be formally labeled or completely integrated into an assurance case. For example, the systems engineering risk management plan and systems safety program plan typically include assurance information and already share some level of integration, but security is not visible within their assurance considerations. The primary value provided by the security version of the assurance case described here is that these existing risk management and safety plans use claims, arguments, and/or evidence in their assurance case.

Security-related system assurance should be integrated into the systems engineering activities to be cost-effective, timely, and consistent. This approach is emphasized due to the emergence of intelligent, proactive, and persistent adversaries, combined with new and increased risks created through networked weapons systems as well as information technology (IT) systems. The assurance case uses an established best practice to integrate security with other disciplines and their designs, verification and validation. It must be understandable, at a high level, by the relevant stakeholders. Figure 2-1 illustrates the typical assurance case framework.

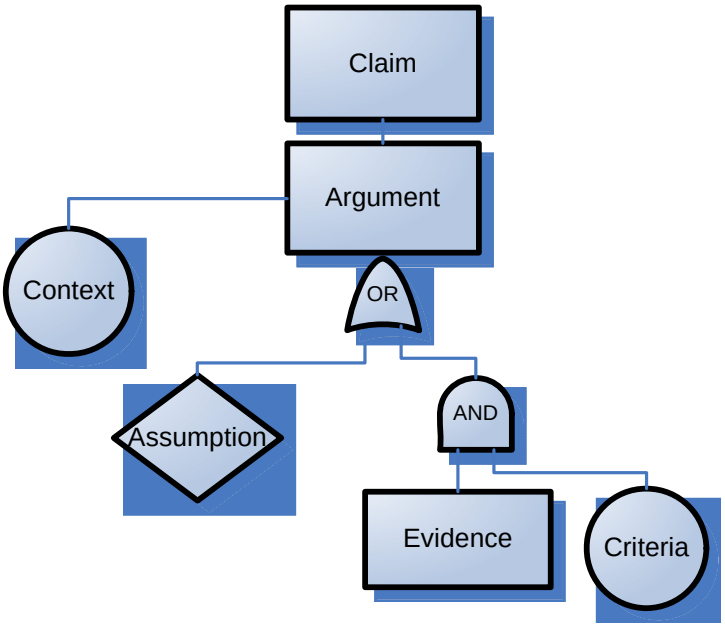


Figure 2-1 Assurance Case Framework

Different disciplines use the same word with different meanings, especially terms such as "assurance," "integrity," and "dependability." (See the glossary for definitions used in this guide.) For example, a system may be required to implement a function controlling hazardous processes. Yet that system may also include anti-tamper (AT) mechanisms that protect this function by destroying the element when it is tampered with. Developing an assurance case can reveal that the AT mechanisms would disable control of the hazardous processes, and aid in ensuring that both requirements are met (e.g., the AT mechanism would need to enable a safe shutdown process).

For more information on including software-related issues in an assurance case, see the Software Assurance Common Body of Knowledge (SwACBK) (Redwine 2006). At the time of this writing, more detail and guidance via international and national standards for assurance cases exist or are under development (ISO/IEC 15026 and guidance for DEF-STAN-056). Another example of potentially useful information is the U.K. Ministry of Defense SafSec guidance (SafSec: Integration of Safety and Security) and standard (SafSec Methodology: Standard).

2.2.1 Claims

Claims identify the system's critical requirements for assurance, including the maximum level of uncertainty permitted for them. A claim must be a clear statement that can be shown to be true or false–not an action. Claims should be precise and clear to the relevant stakeholders. Claims may be initially identified during the requirements identification process, by contract, or by other means. Claims are often initially identified by stakeholders including users, suppliers, and system integrators. Within the U.S. government, information assurance claims and requirements are governed by information assurance controls; at the time of this writing, the civilian sector, DoD, and the IC are working toward a common set of controls.

Claims are usually rendered into sub-claims to help simplify the argument or evidence needed to support the claim (see Figure 2-2). For example, a claim might be that "unauthorized users cannot gain control over the system"; this claim might be subdivided into sub-claims that such control cannot be gained physically, through a network, through subversion of a system element's supply chain, or by social engineering. Claims might be subdivided by normal and off-normal functionality, or by different portions of the architecture.

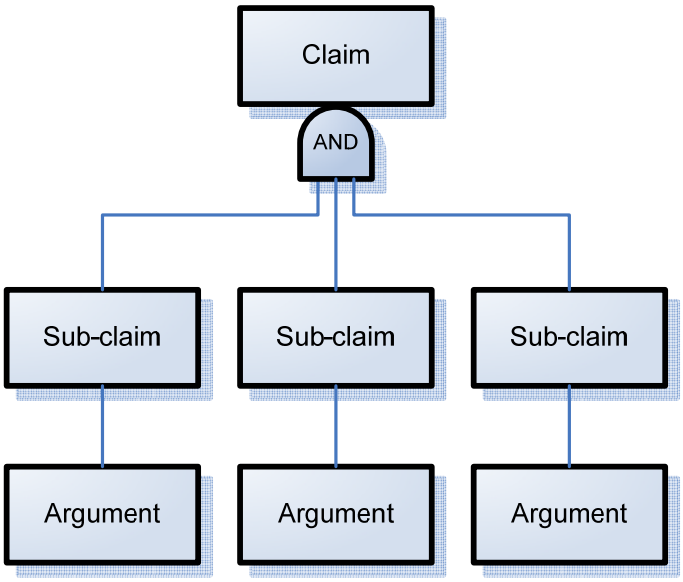


Figure 2-2 Assurance Claim Construct

Claims should identify the required level of confidence. For example, claims should identify if the claim needs to be made high assurance (comprehensive evidence that the claim is met, such as through mathematical proofs and similar means), or merely medium assurance (significant effort is made to reduce the likelihood of failure, e.g., using tools and techniques that significantly reduce the likelihood of weaknesses and vulnerabilities).

2.2.2 Arguments

An argument in an assurance case is a justification that a given claim (or sub-claim) is true or false. The argument includes context, criteria, assumptions, and evidence. Arguments link the SA claim to the evidence. The initial assurance case, except for pre-existing evidence, will generally not have actual evidence but will propose what evidence must be produced to justify the argument that the claim is met. Thus, the initial assurance case allows management to plan and justify system life cycle tasks and postulate what effect(s) might result from not providing certain evidence.

Arguments should be clear, consistent, well-reasoned, and complete (e.g., cover the entire claim or sub-claim). Arguments will typically decompose claims or sub-claims into lower and lower level arguments that eventually connect to assumptions and evidence (see Figure 2-3). The argument will include context (the environment or conditions under which it is effective), which is shown as a separate component in Figure 2-3 to emphasize its importance. The sub-arguments

of an argument are themselves arguments. To build an argument, evidence must meet criteria as discussed below. Some assumptions and evidence may be used in more than one argument.

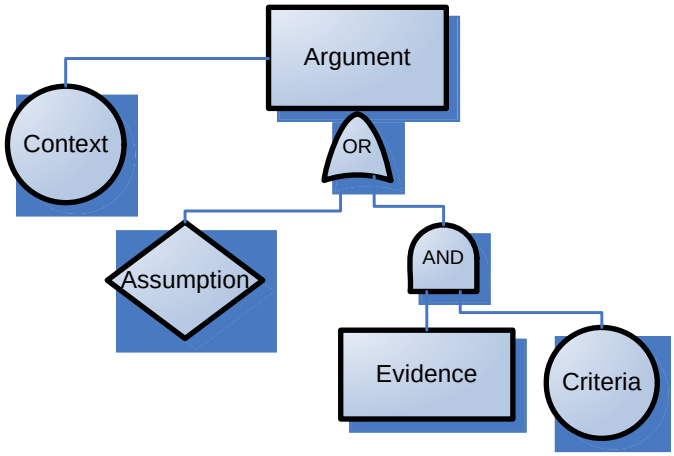


Figure 2-3 Assurance Argument Construct

For rigor, claims, arguments, and evidence should be reviewed at major phase transitions in the system life cycle. Such reviews should ensure that arguments continue to be valid, and that any unnecessary and/or duplicate arguments are removed. For example, in the U.S. DoD, these reviews should occur during each Systems Engineering Technical Review (SETR) (see applicable Service and Office of the Secretary of Defense (OSD) guidance).

Arguments are typically broken down into further sub-arguments, sub-sub-arguments, and so on, until they lead to a set of assumptions and evidence. Arguments are often, though not exclusively, shown using a graphical notation; examples include Goal Structuring Notation (GSN), Adelard Safety Case Development (ASCAD), Toulmin structures, and fault trees (which are used in fault-tree analysis). Arguments should be reviewed to ensure that they cover all plausible possibilities. Arguments are typically developed using both top-down and bottom-up approaches:

- The top-down approach breaks the claim down into smaller, more manageable arguments to justify the claim
- The bottom-up approach identifies potential assumptions and evidence that may be used to create the leaves (end points) of the argument
- A gap analysis (comparing the top-down and bottom-up results) may be used to identify what additional assumptions or evidence are needed to support the argument (and thus justify the claim)

An argument is only relevant in context, and criteria determine which evidence may be used. There is always the risk from unknowns (including “unknown unknowns”). Defense-in-depth measures, and wider margins for element performance measures, may assist in surviving through these unknowns.

2.2.2.1 Context

Context explains the environment or conditions (including state or mode) under which the argument is effective. For example, the argument's context may be that the system is operated in an environment where physical attacks are prevented by other means. Context may be viewed as being derived from the aggregate of the assumptions and evidence. Conversely, the assumptions or evidence may be viewed as needing to be sufficient to justify the argument within its context.

2.2.2.2 Criteria

An argument should include criteria for evidence. These criteria will filter the evidence to only evidence that is applicable and adequate for the argument and claim. For example, if an argument requires "thorough testing," then criteria for "thorough" might be "100% functional testing, software branch testing, and fuzz testing." Criteria are used to determine whether certain evidence acceptably supports the argument.

2.2.2.3 Assumptions

Assumptions, combined with evidence, support an argument. An assumption should be justified and verifiable. Assumptions would typically be reviewed and confirmed as early as practicable with stakeholders, and need to be monitored throughout the life cycle to ensure that assumptions continue to be valid.

Assumptions should be justified by some documented rationale (e.g., the installation documentation clearly forbids connection to the Internet and that monthly scans and system audit logs for the existence of connections would detect such connections or that a connection was made). A stakeholder or reviewer would need to review and confirm assumptions. Clearly identifying and labeling assumptions speeds the review process. Identifying assumptions prevents a typical integration error ("I thought you were doing that").

Over time, some assumptions may be validated sufficiently to become arguments supported by evidence. Conversely, key findings disproving an assumption may seriously jeopardize approval of the assurance case, which is why assumptions must be monitored throughout the life cycle. For example, in the U.S. DoD, key assumptions should be evaluated during the SETRs.

2.2.2.4 Evidence

Evidence is information that demonstrably justifies the arguments.

To be adequate:

- Evidence must be sufficient for its use in the assurance case arguments, including both its quality and its provenance.
- The stated context and criteria apply to each piece of evidence.
- Where inputs, states, or condition can vary, evidence must cover all possibilities or have a sufficient sample to justify the argument.
- Evidence must be uniquely identified so that arguments can uniquely reference the evidence.
- Evidence must be verifiable and auditable.
- Evidence should be protected and controlled by the configuration management (CM) process.

Sample Evidence

- Hardening of development environment (e.g., access controls and identification and authentication (I&A) for CM)
- Development tools that prevent weaknesses/vulnerabilities (e.g., safe languages, safe language subsets, secure libraries, etc.)
- Certification of trusted foundry/trusted supplier
- Static analysis tool results for design and code (e.g., avoidance of vulnerability-prone features, unhandled exceptions, valid/invalid data flow including control of tainted data, conformance to coding standards)
- Mathematical proofs and results of proof-checkers
- Assurance testing (e.g., fuzz testing)
- Modeling and simulation results
- QA results
- Red team results
- Certification of compliance against various specifications, such as National Institute of Standards and Technology (NIST) 800-53, DIACAP, DoDD 8500.01E, and DCID 6-3
- Independent evaluation of a product's security (e.g., Common Criteria, FIPS 140 evaluation, etc.)
- Hardening of system before entering operational environment
- Personnel clearances (of developers, operational administrators, and users)
- Proof of personnel training/education on developing assured systems
- Proof of personnel training/education for operational administrators/users on how to operate securely and respond to incidents
- Verification/test results showing that the target environment's defenses are adequate to protect the system being developed
- Test results demonstrating the adequacy of countermeasures
- Security regression test results (these may be part of a safety regression test, and should occur as part of the sustainment upgrade process before deploying upgrades)
- Development process designed to improve quality (e.g., higher Capability Maturity Model Integration (CMMI) level, large-scale or in-depth review/evaluation/peer review)
- Field history
- Security incidents

Evidence can be used regardless of whether an element is custom-developed, or is a COTS/government off-the-shelf (GOTS) element, though the specific kinds of evidence available will vary. Evidence is collected throughout the life cycle and verified at each review.

2.2.2.5 Building the Assurance Case

Section 3 describes various life cycle processes, and they often generate both arguments and evidence that are necessary to build the assurance case. Subsections named "Building the Assurance Case" are included to emphasize assurance case activities and evidence developed during that part of the life cycle.

3 GENERAL SYSTEM ASSURANCE GUIDANCE

This section follows the top-level outline of ISO/IEC 15288:2008, Systems and software engineering – system life cycle processes, to ease adding system assurance activities to established systems engineering development practices. ISO/IEC 15288:2008 distributes planning and strategy development throughout its processes; to conform to ISO/IEC 15288:2008, this guidebook does the same.

Occasionally, references may be made to the Institute of Electrical and Electronics Engineers (IEEE) 1220, Standard for Application and Management of the Systems Engineering Process (chapter 6), or to the U.S. DoD Defense Acquisition Guidebook (DAG) to provide greater detail about the process being discussed. Table 3-1 maps these documents. Organizations may use other life cycle documents such as EIA 632, Processes for Engineering a System, or ISO/IEC 12207:2008, Systems and software engineering – Software life cycle processes. To use this guidebook, organizations should map their life cycle to ISO/IEC 15288:2008; such mappings are often available.

Table 3-1 Guidebook Correlation to Standards

Guidebook (Section 3)	ISO/IEC 15288 (Section 6)	DAG (Chapter 4)	IEEE 1220 (Section 6)
3.1 Agreement Processes	6.1	Sections 1.2, 2.0.1 – 2.3, 3.0.2 – 3.4, 3.6, 3.7, 4.2 – 4.5, 5.1 – 5.4, 6.4, 7.0.2, 7.2 – 7.4, 7.7 – 7.10	No match
3.2 Organizational Project-Enabling Processes	6.2		No match
3.3 Project Processes	6.3		No match
3.4 Technical Processes	6.4		6.1- .8

3.1 Agreement Processes

The agreement processes (ISO/IEC 15288:2008, Section 6.1) consists of:

- The acquisition process, used by organizations for acquiring products or services from external sources, and
- The supply process, used by organizations for supplying products or services to external sources

A supply chain is the set of organizations, people, activities, information, and resources for creating and moving a product or service (including its subelements) from suppliers through to an organization's customers.

An untrustworthy supply chain can lead to loss of assurance. Suppliers (or suppliers of suppliers) may provide counterfeit parts, tamper with their elements intentionally, provide elements with significant weaknesses and vulnerabilities, and/or reveal confidential information. To prevent this, a clear chain of custody must be identified, approved, maintained, and audited.

The sub-tiers within the supply chain may be critical to the assurance of the final system, so evaluation of the supply chain sub-tiers may be required in both the acquisition and supply processes. The acquisition and supply processes, when assurance requirements are added, affect the system lifecycle of those elements and potentially of their supply chain.

Thus, when performing system-level risk management, consider the elements that make up the system, their supply chain, and the supply chain's impact on assurance.

Today's complex acquisition environment has affected the supply process significantly, with a number of trends that have affected our ability to address assurance needs:

1. Suppliers continue to globalize, establishing overseas activities and outsourcing many parts of the system life cycle including product development, requirements and development efforts, test and evaluation (T&E), support service delivery, manufacturing, etc.
2. It has become cost-prohibitive and untimely to develop systems completely in-house in most circumstances. Systems have become extremely large and complex, requiring the use of many elements developed by various specialists. Leveraging of commercial off-the-shelf (COTS) products and government off-the-shelf (GOTS) products has become accepted practice. The use of commercially available elements (including proprietary and open source software products), standards, and associated development and verification tools has also become the norm.
3. Long acquisition times conflict with the shortened release times of some suppliers. This may force repeated re-evaluation of supplier selection, since decisions made early in the life cycle regarding supply may or may not meet the requirements at the time of the supplier product or service usage. This is especially true in cases where integration efforts may take many years.
4. Technology innovation in many cases is driven by new, small, and/or foreign entities that often have not gone through any supplier evaluation process, making program risk assessment more difficult. Thus, using "best of breed" solutions may introduce significant assurance risks, yet never using "best of breed" solutions may introduce the risk of delivering inferior systems.
   Many of today's products have long operating lives of 20 or more years. During their lifetimes many elements that need to be replaced experience problems with diminishing manufacturing sources. The selection of alternate suppliers or alternate elements invites threats to system assurance.

In support of these trends, the PM and the SE must address the assurance needs of their system while leveraging existing supply management infrastructure. The individuals addressing acquisition and supply processes may not currently have the knowledge to address assurance requirements. They must consider not only the products and services being delivered but also their sub-elements and modifications thereof (some of which may be COTS or GOTS), as well as the tools used to develop and verify the products and services delivered. They should also routinely assess to ensure there is a process for "operational agreement" once the system is live.

In many cases, custom elements (such as application-specific integrated circuits (ASICs) and custom software) may require extra assurance measures. This is because the element developer will typically know or can determine how the element will be used operationally. As a result, they may have the opportunity to divulge confidential information, or to tamper with the element to predict a targeted attack on the system. In addition, weaknesses/vulnerabilities or code tampering are less likely to be noticed and reported by others.

In contrast, an off-the-shelf (OTS) element tends to be developed and tested for generic application. The people in the supply chain of OTS elements will often not know how the element will be integrated into the larger system, and thus are less likely to be able to reveal system-

specific confidential information. However, an OTS element may have serious weaknesses/vulnerabilities, either unintentional or intentional, which can compromise the system in its operational environment.

3.1.1 Acquisition Process

The acquisition process is one in which a product or service is obtained in accordance with the acquirer's requirements. It is used by organizations, including acquisition/procurement offices, systems integrators, and acquirers of many COTS/GOTS elements, for acquiring products or services.

3.1.1.1 Assurance Considerations in Acquisition of COTS & GOTS

Because of cost, time to market, and competitive advantage, systems are increasingly composed of COTS, GOTS, and integration elements. These elements tend to be developed and tested for a generic set of applications as well as large market segments (e.g. banking, federal government, healthcare, etc). The members of the supply chain of OTS elements usually will not know all the possible use cases in which these elements are integrated.

Thus, system development typically involves evaluating OTS alternatives. For assurance, appropriate resources need to be allocated for evaluating the assurance characteristics of the OTS elements. For those OTS elements that are ubiquitous, public information is often available to help determine the likelihood of serious weaknesses/vulnerabilities, and various mitigation options also tend to be available. This information must be considered on a case-by-case basis, to determine which options (custom or OTS) are best for the system being developed. In general, acquisitions should prefer acquiring commercial OTS (COTS) elements instead of building new government OTS (GOTS) elements. In some cases, however, risk analysis may determine that a GOTS approach is best.

Because of the nature of OTS products, the acquirer typically has no influence on current and limited influence on future OTS products. Acquirers should seek OTS solutions that use certified (or best practices) implementations of standards-based interface and should implement a defense-in-depth approach:

- *Standards-based interfaces.* To enable the use of commodity elements, standard interfaces with competing implementations should be evaluated. This evaluation enables switching suppliers, if needed (e.g., because of product weaknesses). In addition, the review process of open standards can remove weaknesses that would otherwise remain in an interface.
- *Defense-in-depth approaches.* Defense-in-depth approaches are the layering on and overlapping of security as well as system assurance measures. In many systems, its resistance to attack is no greater than its weakest link. Using a defense-in-depth strategy, if one defensive measure fails there are other defensive measures in place that should continue to provide protection. Defense-in-depth can be applied inside an OTS element, as well as when combining OTS elements. OTS elements that implement defense-in-depth strategies will tend to be better at countering unintentional weaknesses. Where practical, use two or more levels of defense against primary threats (e.g., suppliers that are least trusted).

In addition, the SE and the PM may need to consider the following factors before choosing a particular OTS product:

- Does the supplier have product evaluation criteria defined for the reduction of potential weaknesses in COTS?
- If a product defect/bug list can be made available, consider examining it. If independent defect/bug lists can be made available, consider examining them too. Do they show that the supplier actively works to prevent weaknesses from being released in their product, and that they place a high priority on repairing important vulnerabilities if found later?
- What are the maturities of the product and the underlying product technologies? An immature product, or a product based on poorly understood technology, is more likely to have vulnerabilities.
- What is the track record/history of the product or organization regarding vulnerabilities? How often do they occur in released products? What were the impacts and how long did they take to be resolved?
- Is it prone to vulnerabilities compared to its competitors?
- Are there publicly known and uncorrected weaknesses or vulnerabilities?
- What is the organization's track record regarding correction of vulnerabilities (including speed and thoroughness)?
- Does the product have certifications that justify its assurance (e.g., Common Criteria evaluation, FIPS 140-2, ICSA evaluation, etc.)? If it has received widespread reviews (including peer reviews or testing), what are their results?
- What mechanisms exist or can be added to prevent the use of counterfeit parts/software?
- What is the supplier's strategy to reduce vulnerabilities?

Note that in some cases, OTS elements should be configured to harden them against attack. For example, the U.S. government released a directive to Federal CIOs that requires new IT System Acquisitions (as of June 30, 2007) operating on Windows XP and Windows Vista to use the Federal Desktop Core Configuration (FDCC), a common secure configuration, and in addition requires providers to validate that the products operate effectively using this secure configuration if they run on Windows XP or Vista (SANS Flash Announcement, March 20, 2007).

3.1.1.2 Assurance Issues in Managing System Integrators

System assurance is not just about assuring the final product; assurance must be addressed throughout the entire acquisition cycle. The system integrator plays a key role in assurance. The acquirer must keep this in mind while managing the integrator.

The acquirer and the system integrator must collaborate to increase assurance throughout the supply chain.

Consider using the following approaches:

- *Limit the information available to the supply chain.*
  - In many cases, the risks from commodity elements can be reduced by preventing the supplier of the product or service from knowing the ultimate user or the use. This approach can reduce the ability of the supply chain to induce a targeted attack. This may be difficult for services or future maintenance of a product (particularly where the Internet or physical address of a customer may give away this information).

- Limiting systems design and operational information reduces the risk of revealing confidential information and also reduces the opportunities for intentional subversion from the supply chain. One method for doing this is to design the system so that the elements with confidential information are separate and can be added after the other elements have been received from the supply chain. For example, a generic SmartCard implementing JavaTM could be purchased from a supply chain, and specific algorithms/applets could then be implemented in-house by trusted personnel (limiting exposure of confidential information). Another example might be to intentionally use FPGAs or microcode to implement certain functions, so that the suppliers provide generic functions and cannot reveal confidential algorithms.

- *Standards-based interfaces*. To enable the use of commodity elements, the use of standard interfaces with competing certified and or best practices implementations should be encouraged. Open standards, in particular, should be preferred, enabling easier switching of suppliers, if needed. (See also Section 3.1.1.1, "Assurance Considerations in Acquisition of COTS & GOTS")
- *Defense-in-depth approaches.* Defense-in-depth approaches are the layering on and overlapping of security as well as system assurance measures. It is assumed that the strength of any system is no greater than its weakest link. Using a defense-in-depth strategy, should one defensive measure fail there are other defensive measures in place that continue to provide protection. Use two or more levels of defense against primary threats (e.g., suppliers that are least trusted).

Identify Suppliers

PMs and SEs should aid the acquisition community in the identification and selection of their suppliers. This becomes even more critical when addressing new technology innovations, products, and service methodologies.

Potential assurance issues include not only unintentional vulnerabilities, but intentional vulnerabilities. Because intentional vulnerabilities can be introduced at various points in the supply chain, protecting the supply chain is crucial.

In some cases, the PM and the SE generally lead the identification of potential suppliers, based on system requirements and market research. At least some of these potential suppliers are typically previously unqualified suppliers (including those who are small or new to the market).

Select a Supplier

Acquirers (including all acquirers in the supply chain) must consider assurance issues when selecting their suppliers. This will often require a better understanding of their supply chain, internal evaluation of suppliers, and/or external supplier evaluation programs (including those for particular market sectors). Selecting a supplier includes consideration of the people, processes, products/services, and the tools. System assurance requirements for services (e.g., systems integration) are addressed in each of the elements discussed below in support of selecting a supplier.

Suppliers must be considered together with an assessment of their system assurance risks. It is impractical to evaluate in depth all potential risks for all suppliers, so the risks must be prioritized. Generally, careful assessment should be applied to suppliers of key elements,

especially custom elements, and/or suppliers of elements whose use is so widespread that it is difficult to ensure that their failure would not cause enterprise-wide failure.

PMs and SEs must consider that the supply chain includes not only OTS products, but also the integrators that use such products to develop systems and systems of systems. For supply chain assurance considerations, the people, processes, products, and tools must be addressed to provide a comprehensive evaluation.

People

Although there is limited control over the various organizations and their personnel in the supply chain, consider assessing their reputation and qualifications, as relevant.

- *Reputation.* Does the organization have a reputation for providing authentic elements (as opposed to counterfeit parts)? Does it have a reputation for producing/providing quality elements? Does it know SA or functional requirements of the system or acquisition processes? Is it a qualified supplier (e.g., an authorized distributor, etc.)?
- *Qualifications.* Do the personnel/organizations used in the development of the products have the knowledge (possibly reflected through training or certifications) of security engineering, assurance methodologies, and the integration of them across the full life cycle, in association with their tasks? (Without this knowledge, they may inadvertently create weaknesses in the system.)

Processes

Current processes do not always adequately address system assurance. However, existing life cycle processes such as evaluation and certification processes provide a framework for system assurance, and for the collection of evidence for system assurance.

Consider process questions that may enhance assurance as noted below:

- What system life cycle processes are implemented and what certifications achieved by the supplier that relate to assurance or security engineering practices (e.g., tool-based evaluation of source or binary code, peer review, verification of defensive functions, red teams, etc.)? (See Sections 3.3 and 3.4, Project Processes and Technical Processes, for more detail.)
- How rigorously are they practiced?
- Are there process controls in place to prevent or limit subversion (e.g., configuration management)?
- Will these processes produce predictably assured outcomes?

Examples of process evaluation approaches, or specifications that imply process requirements that can be leveraged for assurance, include ISO 9000, ISO/IEC 27001, Six Sigma programs, CMMI, SSE-CMM, RTCA/DO-178B, HIPAA, SOX, The National Information Assurance Partnership (NIAP)'s Common Criteria Evaluation and Validation Scheme (CCEVS) Program, etc. Supplier-performed processes are typically ineffective for countering the risk of a malicious organization, but they may help counter malicious individuals or subcontractors, and unintentional vulnerabilities.

Products/Services

Assurance requirements for acquisition of products are discussed above (3.1.1.1). When acquiring a service, consider reputation, qualification, and proven processes as discussed above.

Tools

Tools have become so critical to implementing the system life cycle that they can intentionally or unintentionally destroy the assurance of a system, and do so in a way that is unobserved by the system's developers. Ken Thompson's "Reflections on Trusting Trust" (Thompson 1984) demonstrated that subverting one tool, a compiler, could subvert an entire system. In Thompson's example, the malicious compiler inserted a Trojan horse, changing the system's function and creating vulnerabilities. Today many tools are used in the development of hardware and software, and these tools can access the system under development in ways that could result in intentional or unintentional vulnerabilities in a system.

Development tools may introduce vulnerabilities, such as Trojan horses, or buffer overflows into the system. Such tools include source code generators (including model-driven architecture approaches), software compilers, assemblers, hardware compilers (e.g., VHDL), and tools to correct integrated circuit masks. A tool should have passed qualification requirements for use if it eliminates, reduces, or automates a process step without its output being verified. Where possible, opt for tools that are deterministic (i.e., they deliver the same output for the same input in the same environment), since these are much easier to qualify and test. Spot-checking of results, or even careful examination of generated results, may be appropriate in some circumstances.

Some verification tools may not be able to introduce vulnerabilities, and exacerbate the failure to detect errors or weaknesses that they are expected to detect. These include such tools as static analysis tools that automate a system verification process activity – source and binary code analysis focused on development of both intrinsic and defensive function implementations. Other verification tools include type checkers and functional test tools that focus on evaluating elements. Such tools should be used in an optimized manner so as they are able to detect as many vulnerabilities as possible in the system (e.g., not allowing them to modify the "real" system elements). Tools and/or tool vendors may undergo qualification, and/or go through supplier evaluation processes. The limits of such tools must be understood by their users. Innovation is ongoing in this area, so tool users will need to keep up with changes in the industry.

From acquisition through usage of tools, it is important to ensure that the tools are not tampered with or are not introducing errors or weaknesses into the system that can reduce system assurance levels.

Building the Assurance Case

Evaluate element suppliers, and select elements based on their ability to support the assurance case, to justify the assurance claims against the system threats. Off-the-shelf element suppliers typically cannot add new information; but they can often provide some evidence to increase confidence in their element. System threats and assurance claims should be used as the basis for deciding on the elements to be supplied and the associated evidence needed for the assurance case. When evaluating element suppliers, consider the risks of both intentional and unintentional vulnerabilities being included in that element, and if there are mitigation approaches available.

When building the assurance case, possible evidence (where * indicates evidence that may help against intentional vulnerabilities) includes: supplier reputation*, track record (defect and vulnerability reports, including uncorrected vulnerabilities), past speed of correction, standard interfaces with open documentation of extensions, transparency of implementation*, CM processes that protect the software (e.g., multi-person review)*, product assessment, and compensating mechanisms. Product assessment can include tool-based assessment (using static and/or dynamic tools), peer reviews, and certifications (such as Common Criteria); such assessments may be done by the acquirer, the supplier, or an independent third party. Compensating mechanisms may include limiting the amount of information available to the supply chain*, defense-in-depth, development processes (e.g., ISO 9000, CMMI, SSE-CMM), patch management processes, and individual qualifications.

Any of this information provided by the supplier should be provided in writing with the appropriate approval. Once a supplier has been selected, the rationale and supporting evidence should be documented so it can be traceable, and so that if issues arise or changes are required, they can be quickly addressed.

3.1.2 Supply Process

The ISO/IEC 15288:2008 supply process focuses on providing an acquirer with a product or service in accordance with the acquirer's requirements. The supply process is used by organizations such as systems integrators and product suppliers. For information on how to work within the framework of assurance in the systems life cycle process, review Sections 3.1, 3.3, and 3.4 (Acquisition, Project, and Technical Processes) for specific guidance. When acquiring sub-elements within a system from other organizations, see Section 3.1.1, Acquisition Process. System delivery must ensure that the intended product is received; see Section 3.4.7, Transition, for more detail.

3.2 Organizational Project-Enabling Processes

Organizational Project-Enabling Processes (ISO/IEC 15288:2008, Section 6.2) support the organization's capability to acquire and supply system products or services through the initiation, support, and control of projects. They provide the resources and infrastructure necessary to support projects (system and SoS life cycles) and ensure the satisfaction of organizational objectives and established agreements.

The organization has policy, governance, and technical constraints that affect the systems being developed. This document focuses on the system life cycle, and not the organizational project-enabling processes. System life cycle processes are connected to organizational processes. For example, the requirements analysis and architectural design processes should be enabled by and should capture the requirements and constraints of the environment in which the system will reside, including any policy, governance (including Human Resources), and technical constraints.

3.3 Project Processes

Project Management (PM) with support from Systems Engineering (SE) is critical to the development or acquisition of an assured system. To deliver system assurance, the PM must consider the mission, threats against the mission, system risks, and their relationship with system

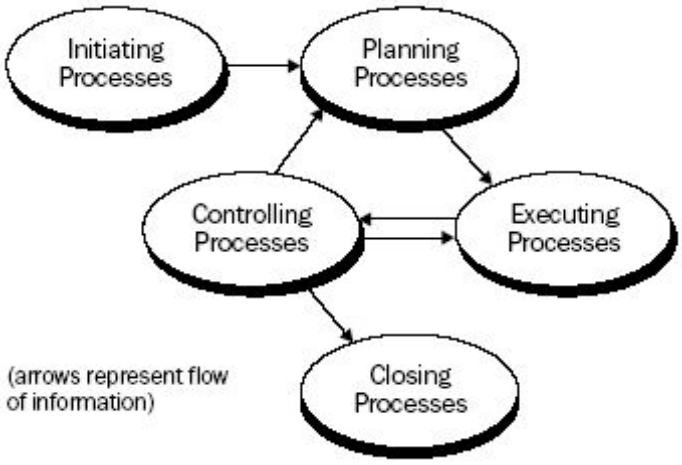
assurance. The following subsections augment existing PM practices with assurance guidance to manage system risk, allowing the PM to balance the necessary rigor to achieve assurance versus project cost and schedule.

As with the rest of the guidebook, this section mirrors the ISO/IEC 15288:2008 organization, referencing additional PM and SE processes (ISO/IEC 15288:2008, Section 6.3). Table 3-2 compares SA Guidebook sections with the ISO/IEC 15288:2008, the Project Management Book of Knowledge (PMBOK 2004), and International Council on Systems Engineering (INCOSE) processes.

Table 3-2 Guidebook Project Processes Mapped to ISO/IEC 15288, PMBOK, and INCOSE

Guidebook Subsections (of Section 3.3)	ISO/IEC 15288 Project Processes (Section 6.3)	PMBOK Processes	INCOSE SE Processes
3.3.1 Project Planning	6.3.1	Initializing & Planning	Technical Management
3.3.2 Project Assessment	6.3.2	Controlling	Technical Management
3.3.3 Project Control	6.3.2	Controlling	Technical Management
3.3.4 Decision Management	6.3.3	Controlling	Technical Management
3.3.5 Risk Management	6.3.4	Planning	Technical Management
3.3.6 Configuration Management	6.3.5	Executing	Technical Management
3.3.7 Information Management	6.3.6	Executing	Technical Management

Figure 3-2 illustrates PM processes according to L. Rowland:



Source: L. Rowland, Hawaii Pacific University

Figure 3-1 Program Management Processes

3.3.1 Project Planning

Project planning (ISO/IEC 15288:2008, Section 6.3.1) is the process used to produce and communicate effective and workable project plans. The project tasks, deliverables, resources, and schedules required to implement system assurance must be incorporated into these plans. Note that the project planning processes of OTS elements are addressed by the OTS suppliers.

3.3.1.1 Define project objectives, constraints, and scope

Ensure system objectives and constraints include assurance as it relates to project performance, quality, cost, and stakeholder needs.

3.3.1.2 Establish a work breakdown structure

The WBS should identify the key operational and/or mission-critical subsystems and elements. These are subsystems or elements that need to undergo greater scrutiny during system development and integration to ensure mission completion while maintaining the confidentiality, integrity, availability, authentication, accountability (including non-repudiation), and auditability of the system. Regardless of how the assurance case documentation is implemented, there must be a way to identify all the assurance claims, and from those claims trace through to their supporting arguments, and from those arguments to the supporting evidence.

The WBS should include:

- Line items to ensure that system assurance is part of all project processes, for example:
  - Program and system threat assessment, including threat scenarios
  - Assurance case development and maintenance (see Section 2.2, Assurance Case)
  - Peer reviews of design and implementation focused on assurance issues
  - Static analysis during implementation
    - Source code analysis during implementation and integration
    - Binary analysis for elements for which source is unavailable
  - Implementing countermeasures (e.g., anti-tamper)
  - Penetration test/red team activities
- Tasks required to assure non-developmental items (NDI). To achieve the required assurance level for a system, some NDI may require additional assurance efforts.
- Risk management activities related to assurance to address weaknesses, vulnerabilities, and evolving threats, and their impact to the system.

(For software issues in the WBS, see also Fedchak, McGibbon, and Vienneau 2007).

3.3.1.3 Define and maintain a project schedule

Assurance case development and maintenance must be part of the project schedule.

3.3.1.4 Define project achievement criteria throughout life cycle

Ensure that major decision gates require passing measurable assurance criteria. Often these will be existing criteria or measures. Examples of such criteria include:

- Vetted list of potential threats and mitigations

- Design and implementation peer review report highlighting significant assurance-related concerns with associated mitigations
- Results of static analysis, both automated and manual, highlighting resolution of source code and binary analyses issues
- Analysis and resolution of certification and accreditation test results
- Analysis and resolution of Common Criteria evaluation results
- System test results that include assurance-related tests (such as testing system input validation to demonstrate rejection of sample malicious data) and resolutions
- Analysis of System Risk Assessment and hardening results
- Analysis of red team results (e.g. system penetration testing) highlighting resolution/actions to address high/medium risks

3.3.1.5 Define project costs and plan a budget

Ensure that adequate funds are allocated to assurance activities. Such funds should be based on the project schedule, labor estimates, infrastructure costs, and so on. The risk management process considers the assurance risks of a system; the tasks to mitigate those risks must be adequately funded in the budget for the mitigation activities to be effective.

In some cases, some elements may be so critical that high assurance measures must be taken (e.g., formal methods). If high assurance measures must be used, budget accordingly; these measures may cost more in time and money. Where a red team is used, funding should be allocated for addressing problems found by the red team, and not just to perform the red team analysis itself. Such project planning and budgeting tools as COSECMO (Cost Security Model) and COCOMO (Cost Control Model) may be used in support of this effort (Colbert 2006).

3.3.1.6 Establish structure of authorities and responsibilities

Ensure that those tasks requiring assurance expertise are identified and that they will be performed by qualified individuals. Assurance impacts all aspects of systems life cycle as well as many key roles such as architects, developers, program managers, systems engineers, contracts personnel, system administrators, and others. Identifying the core set of roles that can influence, resource, and shape the assurance of any system/element results in more affectively achieving its targeted system assurance.

3.3.1.7 Define infrastructure and services

Identify the infrastructure and services necessary for system assurance. Ensure that the facilities and communications networks will provide the necessary confidentiality, integrity, availability, authentication, accountability (including non-repudiation), and auditability, and are adequately protected from attack. This includes supporting all necessary access controls. Ensure that IT assets (including CM tools and processes) are adequate for the system life cycle before they are used, e.g., CM tools should be able to identify exactly who made a change, what the change was, and when it occurred (see Section 3.3.6, Configuration Management). Identify changes in policy necessary to address assurance.

3.3.1.8 Plan acquisition of materials

Develop a detailed plan to manage the risks from vulnerabilities and weaknesses that may impact assurance goals for the system. Existing supplier assurance, supplier selection, and/or assured acquisition processes, should be used as part of the decision process regarding COTS/GOTS elements, particularly for a system of unknown pedigree (SOUP). (See also Section 3.3.1.5, "Define project costs and plan a budget," and Section 3.1, Agreement Process, for more information.)

3.3.1.9 Generate and communicate a plan for technical management

While communicating the plan, ensure that appropriate safeguards are put in place by the technical management team to support assurance requirements.

3.3.1.10 Define project measures

See the assurance case process section for more information on generating, collecting, and using relevant assurance information. For system requirements, see the requirements analysis process.

3.3.2 Project Assessment

The project assessment (ISO/IEC 15288:2008, Project assessment and control, Section 6.3.2) process determines the status of the project. A plan may need to be modified, based on the results of an assessment.

For system assurance:

- Determine the variance between planned and actual results, including the variance's impact on system assurance
- Focus on assessment of elements where there are indicators of potential assurance issues. For example:
  - critical elements with unusually high change rates or fault densities
  - significant changes in supply chain, including change in supplier
  - systems and elements of high complexity
- The assurance case should be revisited when considering a new technology, process, or supplier, to ensure that the technology or process does not enable a new threat or undermine the safeguards already in place to protect the system

3.3.3 Project Control

Project control (ISO/IEC 15288:2008, Project assessment and control section 6.3.2) refers to the direction of the project plan execution and ensures that the project performs according to plans and schedules, within projected budgets, while satisfying technical objectives.

Ensure that corrective and preventive actions, which on the surface might not appear to be related to assurance, do not adversely affect system assurance. In some cases an action that appears to be beneficial can, as a side effect, cause serious impact to the assurance of the system. Examine the assumptions in the assurance case to counter this.

Ensure that corrective actions important to assurance are actually addressed, resolved, and deployed. Customers often complain when a system fails to implement critical functionality, but often do not complain about a vulnerability until it is being actively exploited (which may be too late). When an issue arises in an acquired good or service, including a functional defect, vulnerability, or weakness, constructively interact with the suppliers to correct it.

3.3.4 Decision Management

Decision Management (ISO/IEC 15288:2008, Section 6.3.3) refers to the practice of selecting the most beneficial course of project action. See guidebook sections 3.3.5, Risk Management and 3.4.3, Architectural Design. These processes include considering available options, conducting effectiveness assessments, and making technical trade-offs during the decision-making process.

System assurance needs to be actively addressed as part of the risk management process. Any tradeoff decisions must consider the threat posed by the active adversary. System assurance needs to be a documented requirement, included in any decision-making process.

For each decision situation, the impact of the proposed alternative action and its associated risks must be evaluated, using the assurance case to ensure that the overall system assurance has not been compromised. Once a decision relevant to assurance has been made, its impact must be reflected in the assurance case.

3.3.5 Risk Management

Risk management (ISO/IEC 15288:2008, Section 6.3.4) attempts to reduce the effects of uncertain events that may result in changes to quality, cost, schedule, or technical characteristics. System assurance issues are often not considered during the risk management processes, activities, and work products, resulting in project stakeholders unknowingly accepting assurance risks that can have severe financial, legal, and national security implications. Risk management covers the entire life cycle, including the operation and maintenance processes. A PM, with assistance from an SE, must realize that a system assurance threat is a threat to program performance. Through risk management, risks (including system assurance risks) are identified and categorized, their probabilities and consequences are determined, a risk mitigation strategy is specified for each risk, the risk status is communicated, and unacceptable risks are acted upon.

Upon determining the organization's implementation of risk methods and policy, the existing PM and SE processes for risk management need to be reviewed and expanded to address system assurance risks throughout the system life cycle. This section addresses those risks. Generic risk management is not addressed here.

In the execution of the program, the PM and SE will need to follow the plan for protecting the system against threats to the assurance goals. As part of this process, the PM and SE will have to reevaluate and update, as needed, the assessment threats, the weakness and vulnerabilities, and any other changes that affect the overall risk mitigation plan (such as costs).

3.3.5.1 Establish risk management strategy

A risk management strategy is a systemic approach to risk identification, assessment, and mitigation. There must be a continuous and iterative system assurance process that delivers due diligence for identifying and assessing risk issues, as well as mitigating issues that are selected as

requiring attention. The assurance process, including risk management, is integrated throughout the life cycle, including during operations and maintenance, due to emerging threats and the dynamic nature of today's operational environments. There is a risk of subversion throughout any part of the system life cycle processes, including risk management itself. Consider using the assurance case's claims, arguments, and evidence as a framework for organizing and addressing system assurance risks.

PMs must assume the existence of an intelligent and malicious adversary. Thus, probabilities must be measured differently than for naturally occurring events, as they may not be accurately computable, and they may not even be applicable. Normally, probabilities can be used to accurately represent the likelihood of a natural event occurring. However, an intelligent adversary can react to countermeasures, and choose the time and place to make an attack. Depending on the system, a security breach can have a catastrophic effect on safety. Moreover, highly motivated and well-resourced adversaries must be assumed.

At a minimum, security engineering must provide safety engineering with information about potential security breaches and their probabilities (if computable). Even better would be to develop the security and safety cases jointly to ensure that conflicts between safety and security are resolved early in the development cycle, and that it is possible to take advantage of commonality between safety and security for risk management.

There is a direct relationship between the requirements analysis process and risk management. The requirements analysis process identifies system assurance requirements (among other requirements). The risks of failing to identify or implement such requirements are covered here, in risk management. Note also that the risk management processes impact all other processes.

3.3.5.2 Identify and define the risks

The risk management process must consider the risks on system assurance from the supply/acquisition chain, the system life cycle personnel (including the developers, integrators, operators, and maintainers), and other threat agents. Consider the risks of insertion of both unintentional and intentional vulnerabilities, from both trusted and untrusted sources, throughout the life cycle (including system development, operations, maintenance, and disposal).

Identification of risks should be an open and encouraged process. While the process may lead to an increase in identified risks, the processes to assess legitimacy and prioritize the risks should be just as robust, thereby leading to a manageably-sized database for further mitigation activities. To identify and define risks, programs should consider using repositories of threats, weaknesses, and vulnerabilities, including such sources as:

- US CERT (http://www.us-cert.gov/) Technical Bulletins and Advisories
- Common Vulnerabilities and Exposures (CVE®): http://cve.mitre.org/
  This site provides a list of the standard names for security vulnerabilities and exposures.
- Common Weakness Enumeration (CWE): http://cwe.mitre.org/
  This site provides a dictionary of common software weaknesses. CVEs are instances of CWEs.
- Common Attack Pattern Enumeration and Classification (CAPEC): http://capec.mitre.org/

This site provides a dictionary of patterns that malefactors could use to attack computer systems.

The PM/SE should also consider the specific system requirements including operational environment that may affect the use in the system assurance case.

Identification and definition of risk should include circumstances or events that can potentially harm the system through destruction, disclosure, or modification of data and/or denial of service. Such results can be caused by unintentional or intentional vulnerabilities introduced by authorized system creators, as well as unauthorized individuals and organizations gaining control over the development environment. For example:

1. A threat agent, masquerading as an authorized user, gains access or greater privilege to a system through stolen logon IDs and passwords, impacting system confidentiality:
2. The threat agent may insert malicious code to impact the system operations in many ways.
3. The threat agent may alter data, in transit, impacting data integrity. These data can be reordered, deleted, or modified through the use of such simple techniques as sniffing and replay.
4. A threat agent prevents a system from functioning in accordance with its intended purpose, creating a denial of service attack, impacting system availability. This attack may be achieved through the subversion of a piece of equipment, rendering it inoperable and/or forcing it to operate in a degraded state using a time-delay attack.

As the risks are identified, they should be collected into a risk registry (a list of risks and information about them) for analysis. Developing the risk registry should begin early in the system development process and be a continuing effort.

In general, one risk to be identified and addressed is whether there is a potential for not being able to achieve the necessary system assurance in a timely manner, potentially resulting in a need to risk mitigate the system certification or accreditation, and/or resulting in the system not being used as intended. The use of this guidebook is intended to help reduce this risk, and should be included as part of the risk mitigation plan.

3.3.5.3 Determine the probability and possible consequences

The probability and possible consequences associated with each risk occurrence should be determined. Note, however, that quantitative probabilities are difficult to pre-determine for system assurance. One reason is that intelligent adversaries tend to attack those areas of the system or its development processes that are least defended. If only areas with a high probability of receiving attack are hardened against attack, the adversary is more likely to attack the unhardened areas—which had previously been considered as having lower probability of receiving an attack. In addition, methods for obtaining measures of system assurance are still in their infancy, though there is ongoing work to improve this. Thus, system assurance probabilities will often need to be determined qualitatively (e.g., high/medium/low). In addition, these probabilities will need to be revisited on a situational basis, to consider environment changes or if an intelligent adversary might exploit them as “easier” attack paths. Thus, risk management results need to be less sensitive to the accuracies of the probabilities, typically using a layered set of approaches as discussed below.

Where judging risks is difficult, one possible approach is to make them at least not unacceptable (tolerable) and "as low as reasonably practicable" (ALARP). In employing the ALARP approach, judgments about what to do are based on the cost-benefit of techniques – not on total budget. However, additional techniques or efforts are not employed once risks have achieved acceptable (not just tolerable) levels. This approach is attractive from an engineering viewpoint, but the amount of benefit cannot always be adequately established.

When considering the possibility of supply chain subversion, both intentional and unintentional, consider the risks associated with that supplier (prime, subcontractor, and so on), its infrastructure (e.g., network, configuration or product data management system), and its personnel. These subversions may result from internal or external threats on that supplier. Supply chains quickly become complex, since suppliers have suppliers, which then have suppliers, and so on. See Section 3.1.1, Acquisition, for more information.

Risk management must consider the broader consequences of a failure (e.g., to the mission, people's lives, property, business services, or reputation), not just the failure of the system to operate as intended. In many circumstances, consequences can vary probabilistically. However, in system assurance, an intelligent adversary can choose its attack to occur at the worst possible time, with the worst possible consequences. An intelligent adversary can even choose to cause multiple simultaneous failures, some of which may be in the environment. Thus, for system assurance, the entire chain of following events must be identified and considered to determine the worst possible consequences. For example, loss of a critical service such as traffic control system failure of a major metropolitan area could result in major accidents, not just a traffic jam. A critical infrastructure control system that shuts down electrical generation could cause not just regional blackouts but deaths due to loss of heat/cooling, and massive property damage due to flood pump failure. Thus, the system assurance case should be sufficiently broad so as to identify not just system (including SoS) failures, but the larger consequences of those system failures such as mission failure, loss of life, physical harm, and major loss of property.

One possible consequence is the reduction in the assurance of the system itself. For example, the system assurance may have been predicated on certain assumptions later found to be false (e.g., due to a breach). The system assurance case arguments should aid in determining the impact and methods for remediation, should this occur.

3.3.5.4 Evaluate and prioritize the risks

The risks should be evaluated and prioritized in terms of probability and consequence. As noted above, prioritization needs to be less sensitive to the accuracies of the probabilities, and consequences must consider the chain of effects that an intelligent adversary could cause.

3.3.5.5 Determine the risk mitigation strategies

For each risk, determine the risk mitigation strategy, including risk avoidance, transfer, mitigation, acceptance (retention), and/or some combination. Consider trade-offs across the enterprise and mission, as well as trade-offs between system risks, balancing between them and the resources available.

In many cases, a single defense mechanism will not be sufficient to mitigate risk. Instead, layered defenses (e.g., defense-in-depth or engineering-in-depth) may be necessary to provide

acceptable risk mitigation. Some risks can be avoided or eliminated, for example by changing the software's design, elements, or configuration, or the configuration of its environment. Some approaches may involve a trade-off between performance and security. These trade-offs may be easier to perform by leveraging element(s) threat models, the element's associated risks and its role in the overall system mitigation strategy. (See also Sections 3.4.3, Architectural Design and 3.4.4, Implementation.)

3.3.5.6 Define the threshold of acceptability for each risk, and actions if exceeded

Different stakeholders may have different thresholds of acceptable risk. In some cases, degradation of service to a certain amount may be acceptable.

3.3.5.7 Communicate and maintain the risk registry, risk mitigation actions, and their status

The risk registry information, mitigation actions, and their status may need to be protected from disclosure, since this information could be a significant aid to an attacker. Risk registry is also key to the building of the assurance case as it can contain critical information such as claims, arguments as well as the evidentiary data and other relevant information in support of the assurance case.

The amount of information required for a given risk may increase depending on the criticality level of the system, the importance of that risk, and the stringency of the system's certification and accreditation process. Risk management data, generated as a normal part of a product/service development, may be provided to C&A officials as augmenting evidence of IA control implementation, but is not explicitly required for C&A approval. The verification process may provide some of the objective evidence to justify that a given risk has been managed.

3.3.5.8 Methods for assessing security risk

Several system and software security risk assessment methods exist, some with supporting toolsets:

- Microsoft's threat modeling as described in Swiderski and Snyder 2004
- Microsoft's ACE (Application Consulting and Engineering) Threat Analysis and Modeling (Avery and Holmes 2006)
- European Union-funded Consultative Objective Risk Analysis System (CORAS) and Research Council of Norway-funded Model-Driven Development and Analysis of Secure Information Systems (SECURIS) (Hogganvik and Stølen 2006)
- PTA Technologies' Calculative Threat Modeling Methodology (CTMM)
- Trike
- NASA Reducing System Risk program's System Security Assessment Instrument (SSAI)
- CMU SEI's OCTAVE (OCTAVE 2001)
- Siemens' and Insight Consulting's Central Computer and Telecommunications Agency (CCTA) Risk Analysis and Management Method (CRAMM)

While not primarily intended for software, the *Security Considerations for Voice Over IP Systems, Information Security Handbook: A Guide for Managers, and Risk Management Guide*

for Information Technology Systems may also be useful for performing system security risk assessments.

3.3.6 Configuration Management

Configuration management (CM) (ISO/IEC 15288:2008, Section 6.3.5), traditionally a best practice for engineering a high-quality system, plays a key role in establishing one of the most important elements in protecting against exploitation. The key is to employ strict control and detailed auditing of activities related to the ongoing processes. Leveraging such CM resources as tracking databases and spot inventory checks as well as a labeling system(s) that provide specific target tracking, aging, and other sufficient pertinent data to categorize and track all assets should be used. An adversary will find it much more difficult to exploit a system that maintains a strong monitoring and report structure.

Configuration management is a process to establish and maintain the integrity of all identified outputs of a project or process, and make them available to concerned parties. This includes the CM of the information captured for the project as well as the CM used by the supplier for the product as it was being developed. In addition, to support system assurance, the CM process must establish and maintain confidentiality, integrity, availability, authentication, accountability (including non-repudiation), and auditability . It must be rigorous commensurate with the criticality of the system, data, and mission, and be flexible enough to enable addressing a wide variety of threats. CM should be tailored with respect to system assurance, which counters maliciousness, reduces uncertainty, and provides objective evidence. Granularity within the CM system should be adjusted to support the assurance case. In addition, the CM system must use techniques, such as non-repudiation, to deter changes that are intended to subvert the system.

For purposes of this guidebook, the “CM system” is the set of processes, procedures, tools, and so on that implements CM. (CM will not be discussed in general in this document; only system assurance aspects that affect or are affected by CM will be.) A CM system must protect its information, such as the current and any previous configurations, as well as an audit trail of the changes made, who made them, and when they occurred. CM systems must normally keep this audit trail information immutable, to support accountability.

The CM process is interlinked with the information management (IM) process. Here, CM is considered as the framework in which the information is contained. Individual information elements are contained in the framework; collections of information are managed through CM. There may be cases in which information is managed (e.g., distributed) but is directly placed inside a CM system (e.g., status reports, temporary results, e-mails, etc.).

3.3.6.1 Define a configuration management strategy

Define the CM process as it pertains to the system life cycle, including how it will address confidentiality, integrity, availability, authentication, accountability (including non-repudiation),and auditability. Determine the assurance-relevant attributes, the timescales for implementation of the CM process, and the tools required to support them. There may be a need to adjust CM requirements or approach based on what tools and techniques are available.

The CM processes should be maximally automated, particularly for electronically stored items, because it is easier to make unintentional mistakes and to maliciously manipulate manual

processes. Automation provides the additional benefit of reducing the incentive for circumventing the CM process. Note, however, that approaches that can automatically reconfigure systems without appropriate review of those changes, and their negative impact on the system assurance case, should be avoided where practical.

Strategies for CM processes and supporting tools should incorporate the following:

- Use strong per user authentication. If passwords are used for authentication, they should always be encrypted for transmission over a network, and never stored as plain text anywhere.
- Centralized repositories should be hardened from attack. For example, on the platform supporting the centralized repository, limit the number of other services being run to reduce the risk that these other services could expose the repository to attack.
- Restrict and monitor network access for the CM system.
- Establish quality acceptance criteria to avoid introducing inferior code which is easily compromised.
- Audit the CM administration measurements for metrics on access and updates to the data, to determine if there is unexpected or unusual activity (e.g., activity with unusual times, locations, systems, or people; individuals updating an unusually large number of CIs, etc.).
- Maintain configuration control for operational system and sub-system settings.
- Adjust the granularity of the CM system to more easily provide input to the assurance case.

The CM strategy should identify how to determine who can do what, the roles, attributes, and division of responsibilities. Determine the process used to vet/accept elements for certain uses, e.g., where two-person controls and/or approval by configuration control boards is required, to reduce the risk of insertion of malicious elements.

The strategy should discuss how to perform CM of OTS elements, including COTS, and how changes in OTS elements impact system assurance. Determine how OTS versions are identified, and how new patches, bug fixes, and upgraded/new versions of OTS elements are managed. See the acquirer process for information on selecting suppliers.

Identify how to get relevant information from the CM process into the assurance case, including during maintenance.

Patch management's speed should be commensurate with its associated risks. Overly hasty patch management can cause deployed systems to fail to perform their missions and/or insert other vulnerabilities. Conversely, excessive caution in patch management can unnecessarily expose a system to attack by excessively delaying deployment of needed patches. Thus, a CM process for deployed systems must have a triage process to determine which of several patch deployment processes should be used. This triage process should be part of risk management. For example, changes may be fielded immediately or deferred to a later major release.

Systems of systems in which there is an incremental development and deployment model bring in additional complexity. These different systems may have different owners, which results in CM complexities. Should such a situation occur, attempt to establish a relationship with the other entities to define the CM controls for the SoS. CM control may be either hierarchical or

distributed, based on the established relationships. It may require a combination of manual and automated approaches. The CM strategy should encompass any impact on the system assurance, based on interoperability within a SoS context. Determine the strategy and plan for horizontal and vertical CM, in particular, how to identify the configuration items that must be protected on behalf of other systems (within the SoS) and vice versa.

3.3.6.2 Identify items subject to configuration control

Configuration items should be uniquely identified to enable configuration control. Identify the configuration items (CIs) that are critical to system assurance, so that they can be protected as necessary. CIs should include defensive functions, build instructions, documentation, the assurance case, tools (including development and CM tools), and the configuration baselines. Delivered systems should include version information sufficient to identify exactly what version was delivered.

3.3.6.3 Maintain configuration information with integrity and security

Protection of configuration item data and meta-data, both in repositories and under modification, must be maintained. Controlled entry points for those needing access to the CM data should be defined.

Physically protect CM systems (e.g., by locking them up) and control access to the system. Ensure that staff cannot add snooping devices (e.g., keyboard loggers or hardware key loggers), reboot systems to unprotected states, and so on, to gain access.

Establish processes to help counter data loss or subversion of a CM repository. Back up repositories, and automatically compare backups with current versions to detect tampering or data corruption. Consider having developer systems compare centralized repositories with their claimed changes as an integrity check. For example, consider using CM tools that support cryptographically “immutable” chains, so an attacker must create “new” entries (which are more visible) instead of silently modifying old information. Establish a process for reconstructing a repository when subversion has been detected. When subversion occurs, determine root causes, and perform corrective actions to address them, possibly including changes to the CM processes. These are to counter the risk of attacks targeted at the CM repositories, directly or indirectly (e.g., via CM administration functions or other services), whether internal or external.

For OTS CIs, detailed element information may not be available. For example, source code and details about hardware and Internet protocol cores are often considered proprietary information that is not released.

Audits should be used to ensure that the system assurance has been maintained by comparing the expected baseline with the actual deployed system configurations.

3.3.6.4 Ensure changes to configuration baselines are managed

Manage all configuration changes, both of configuration items and of baselines (configurations of configuration items). This includes ensuring that they are properly identified, recorded, evaluated, approved, incorporated, and verified.

To prevent unauthorized changes, the CM system should practice least privilege. For example, changes should be permitted only after considering the role, attribute, or identity of the requestor, along with the configuration item or baseline being modified.

Record every change made, including who made the change, when, and exactly what changed. This information must not be forgeable, and must be protected to support accountability. For example, if it is later determined that a particular developer or supplier is malicious, it must be possible to quickly identify the changes made by that developer or supplier (many CM tools have an "annotation" or "blame" function to support this). As noted earlier, recording this information should be automated as much as practical so that it is easy to use. All changes to each baseline should be traced to the previous baseline(s), and maintained with integrity.

Where possible, report to users "last change from you was (time stamp) from (source location)" information when a new change is submitted, so that users can immediately notice any discrepancy. Track modification requests and changes, especially vulnerability reports, so that any vulnerability can be shown to have been addressed (if it has been).

Attackers can introduce malicious information via a CI that is not managed or controlled by the program or system owner. Programs should be able to identify the suppliers of the system and its elements. In addition, they should have a way of contacting the supplier for each CI to address any possible issues that may arise.

Suppliers of OTS elements may make updates to their elements without providing detailed information about those changes to their customers. These changes can adversely impact assurance. Mitigation approaches should be used to counteract this potential impact. Potential approaches include:

- Developers may enter into agreements with their suppliers to obtain additional information (in particular, for critical elements) where necessary to support system assurance.
- Documentation and audits may be required, where necessary.
- Architectures may be created to mitigate this lack of information (see the architectural design process section).
- Additional verification processes may be used to determine that updated elements meet necessary requirements (see the verification section).
- Require/develop standards/specifications that enforce required attributes.
- Identify suitable substitutes.

As business case and relationship supports, customers/developers may work with their suppliers to obtain additional information (in particular, for critical elements) where necessary to support system assurance.

Update notifications, including those delivering vulnerability repairs, vary from supplier to supplier, depending on their update programs. If the frequency is out of sync with system needs, either too frequent or not frequent enough, it is the responsibility of the PM/SE/System Administrator to monitor the update OTS update program and determine the appropriate policies and procedures for OTS maintenance.

3.3.7 Information Management

Information management (IM) (ISO/IEC 15288:2008, Section 6.3.6) provides relevant, timely, complete, valid, and, if required, confidential information to designated parties during and, as appropriate, after the system life cycle (ISO/IEC 15288:2008). The IM process is interlinked with the CM process. CM is the framework in which individual information elements are contained; collections of information are managed through the framework.

3.3.7.1 Define the items of information that will be managed and the IM strategy

Information is defined broadly and includes information about information (e.g., meta-data). It may require special protection from disclosure, to prevent aiding system compromise by the active adversary. Some information may need to be designated as requiring special protection and may be marked that way.

Marking examples include:

- Proprietary
- Classified (Confidential/Secret/Top Secret)
- Critical Program Information (CPI)
- FOUO (For Official Use Only)

Information examples include:

- System assurance case
- Source code
- Software documentation
- Information about suppliers (particularly those justifying suppliers' trustworthiness)
- Information from suppliers regarding their products/services
- Program plans, such as program protection and the information security plan
- Certification and accreditation (C&A) documents (such as U.S. DoD Information Assurance Certification and Accreditation Process (DIACAP))
- Threats (including raw and analyzed data)
- Anti-tamper techniques being used
- Residual system vulnerabilities
- Bug Reports

The strategy for IM should be coordinated tightly with the CM strategy.

3.3.7.2 Designate authorities and responsibilities regarding origination, generation, capture, archiving, and disposal

Some information that impacts assurance must have controlled and limited access. The designated authorities must be cognizant of their responsibilities to safeguard the confidentiality of such information. Do not forget to identify the authorities required for information maintenance (including archiving and disposal).

3.3.7.3 Define the rights, obligations, and commitments regarding retention, transmission, and access

Access to receive some information may require ensuring that only authorized recipients receive it. More sensitive information, including information that impacts assurance, may need to be encrypted, both when transmitted and when at rest. Such information may need to be authenticated for its integrity and/or nonrepudiation (e.g., with digital signatures). It may be useful to limit the information about the system, its elements, and its configuration, since such information is useful to an attacker (who could use this to identify likely vulnerabilities). Changing some information may require special processes, e.g., two-person controls.

Review the necessity for proprietary, vendor-unique interfaces or formats, which may limit the ability of preventing change or move to another element (e.g., due to copyright, patent, rights). Otherwise, if a serious vulnerability appears in that product, or the supplier stops supporting it, it may be very difficult to switch suppliers. See the agreement process for more information on how to manage suppliers.

3.3.7.4 Define the content, semantics, formats, and medium

Define the encryption algorithms and formats for encrypted data. Consider the issue of key recovery (which permits emergency access to encrypted data), and the trade-off between enabling key recovery and keeping the information confidential. In cases where imprecise semantics could result in system compromise, the use of formal methods to define semantics may be worthwhile. There may be a need for additional filtering mechanisms (and/or outright forbidding) for certain data formats that have a propensity for carrying attacks (e.g., image formats such as graphic interchange format (GIF)).

Protect the physical medium used to store important information (e.g., laptop storage devices, USB sticks, backups). In some cases (particularly for backups) they should be both physically protected and encrypted at rest. For mobile media, consider policies that require filtering information so that sensitive information is not included unless necessary (e.g., blanking out personal information, anti-tamper information, assurance case information, and test results).

3.3.7.5 Obtain the information

Obtain information (commensurate with its importance) using methods that reduce the risk of invalid or manipulated information. Examples include using trusted parties, validating with alternative sources, using digitally signed data (and checking the signatures). Avoid using sources that have an incentive to provide misleading information.

The process of obtaining information (e.g., using search engines with certain keywords, or carefully reviewing specifications on specific products) may intentionally reveal important characteristics of the system to unauthorized parties. Revealing such information could eventually lead to the compromise of such systems. (See also Section 3.3.6, Configuration Management.)

3.3.7.6 Define information maintenance actions

Ensure that any personnel and processes involved in archiving, re-recording, or transforming the information will not modify or redistribute it inappropriately. This is especially important to consider if these actions are done by third parties. In many cases, a system developer or operator

must consider the acquirer's data protection requirements, so that the acquirer's data are not compromised.

3.3.7.7 Retrieve and distribute information as required

Ensure that only the appropriate information is provided to authorized parties. In particular, ensure that providing information of one kind does not accidentally reveal information of another kind. In some cases, information may need to be filtered out or intentionally separated.

3.3.7.8 Provide official documentation as required

Mark official information so that it is difficult to alter, e.g., digitally sign official documents or use holographic seals. Or, at least ensure it is difficult to alter or receive while in transit, e.g., using encrypting and authenticated Web servers (e.g., SSL/TLS, OpenSSH, etc.).

3.3.7.9 Archive designated information

The archival systems themselves must be protected from attack. See also Section 3.3.6, Configuration Management.

3.3.7.10 Dispose of information

Unwanted information may still contain sensitive information, so it should be destroyed at the end of its retention period using appropriate disposal methods (e.g., shredding paper, melting/burning hardware, etc.). For more information on disposal, see ISO/IEC 15288:2008, PMBOK (2004), and INCOSE (2005).

3.4 Technical Processes

This section mirrors the ISO/IEC 15288:2008 Section 6.4, Technical Processes. Table 3-3 maps guidebook subsections against ISO/IEC 15288:2008 Section 6.4, DAG Chapter 4, and IEEE 1220 Section 6. Correlations to other guides and standards will be added as the need arises.

Table 3-3 SA Guidebook Technical Processes Mapped to ISO/IEC 15288, Defense Acquisition Guidebook, and IEEE 1220

Guidebook Subsections (of Section 3.4)	ISO/IEC 15288 Technical Processes (Section 6.4)	Defense Acquisition Guidebook (Chapter 4)	IEEE 1220 (Section 6)
3.4.1 Stakeholder Requirements Definition	6.4.2	4.2.4.1 Requirements Development	6.1 Requirements Analysis
3.4.2 Requirements Analysis	6.4.3	4.2.4.2 Logical Analysis	6.1 Requirements Analysis 6.2 Requirements Validation
3.4.3 Architectural Design	6.4.4	4.2.4.3 Design Solution	6.3 Functional Analysis 6.4 Functional Verification 6.5 Synthesis 6.6 Design Verification 6.7 Systems Analysis
3.4.4 Implementation	6.4.5	4.2.4.4 Implementation	No match
3.4.5 Integration	6.4.6	4.2.4.5 Integration	No match
3.4.6 Verification	6.4.7	4.2.4.6 Verification	6.4 Functional Verification 6.6 Design Verification

Guidebook Subsections (of Section 3.4)	ISO/IEC 15288 Technical Processes (Section 6.4)	Defense Acquisition Guidebook (Chapter 4)	IEEE 1220 (Section 6)
3.4.7 Transition	6.4.8	4.2.4.8 Transition	No match
3.4.8 Validation	6.4.9	4.2.4.7 Validation	6.2 Requirements Validation
3.4.9 Operation (and Training)	6.4.10	4.3.5 Ops & Support	No match
3.4.10 Maintenance	6.4.11	4.3.4 Production & Deployment	No match
3.4.11 Disposal	6.4.12	4.1.3 Total LC Management in SE, 4.2 SE Process, 4.3.5 Operations & Support Phase, 4.4.14 Demilitarization & Disposal	No match

The technical processes (systems engineering process), as noted in the ISO/IEC 15288:2008 standard, is used to:

- Define the requirements for a system.
- Transform the requirements into an effective system.
- Permit consistent reproduction of the system where necessary.
- Use a system to provide the required services.
- Sustain the provisions of those services.
- Dispose of a system when it is retired from service.

The technical processes as they relate to system assurance are tied to the criticality of a given system and its elements. Mission/business need determines the criticality for a system. In particular, what is the effect of the loss or degradation of that system?

A threat leads to a risk. Criticality helps determine the response to the risk. Assurance supports risk management. A notional example of how a system's criticality may be determined is shown in the table below.

Table 3-4 System Criticality

Criticality	Definition	Tolerance	Example
Highest	The compromise of a system or elements within a system results in direct catastrophically degraded mission/business capability of system/mission failures.	Seconds to minutes	Denial of service of an on-line Securities Transaction Systems
High	The compromise of system or elements within a system potentially results in seriously degraded mission/business capability of system.	Minutes to hours	Residential electrical power loss or brown-outs based on an IT compromise
Medium	The compromise of system or element potentially results in partial or identifiable degradation to mission/business capability of system.	Hours or days	Loss of function of traffic lights in a city due to an IT system compromise
Low	The compromise of system or element potentially results in an inconvenience.	Days	

A higher level of criticality requires a higher level of assurance. This distinction must be reflected in the requirements.

3.4.1 Stakeholder Requirements Definition

The purpose of the stakeholder definition process is to define the requirements for a system that can provide the services needed by users and other stakeholders in a defined environment. (ISO/IEC 15288:2008, Section 6.4.2). There is a relationship between the criticality of the system and the level of confidence necessary to achieve system assurance.

The following considerations enhance the requirements process to address system assurance:

1. Identify specific stakeholders or stakeholder classes who have an assurance perspective. In general, stakeholders or stakeholder classes, while they may not express requirements in terms of system assurance, should be able to identify their tolerance for failure, degradation, compromise or loss, e.g., degraded modes of operation.
2. Identify threats, vulnerabilities, or weaknesses. The stakeholders should use current or previous experience in dealing with similar systems to help with the identification. The stakeholders then need to define appropriate requirements to address the threats, vulnerabilities and weaknesses identified.
3. Identify mitigation and/or compliance requirements, such as anti-tamper requirements, FIPS 140-2 compliance, HIPAA, SOX.
4. Consider the operational environment in which the system will reside, and that environment's impact on system assurance, when defining the set of stakeholder requirements.
5. Provide a framework of requirements to validate that the system does what it is defined to do and nothing else, in its intended operational environment.

After the requirements have been collected, the next step is requirements analysis.

3.4.1.1 Building the Assurance Case

See Section 3.4.2.4, Building the Assurance Case section of Requirements Analysis Process.

3.4.2 Requirements Analysis

The purpose of the requirements analysis process (ISO/IEC 15288:2008, Section 6.4.3; IEEE 1220) is to derive measurable technical system requirements specifying the characteristics the system is to possess, based on the elicited stakeholder needs.

3.4.2.1 Define the system functional boundary

Clearly define the system boundary, and in particular identify what is and is not in the scope of the system, based on stakeholder needs. To develop defense-in-depth, individual systems need to implement assurance in coordination with protection mechanisms that are provided within its supporting infrastructure. The system will leverage many services from the environment, and for assurance it is important to identify which services are being leveraged vs. which services are being provided by the system itself. For example, does the system leverage existing cryptographic functions, operating systems, hardware platforms, I&A mechanisms, audit, and/or other network security services? Or, is the system providing them? For assurance purposes, the system must

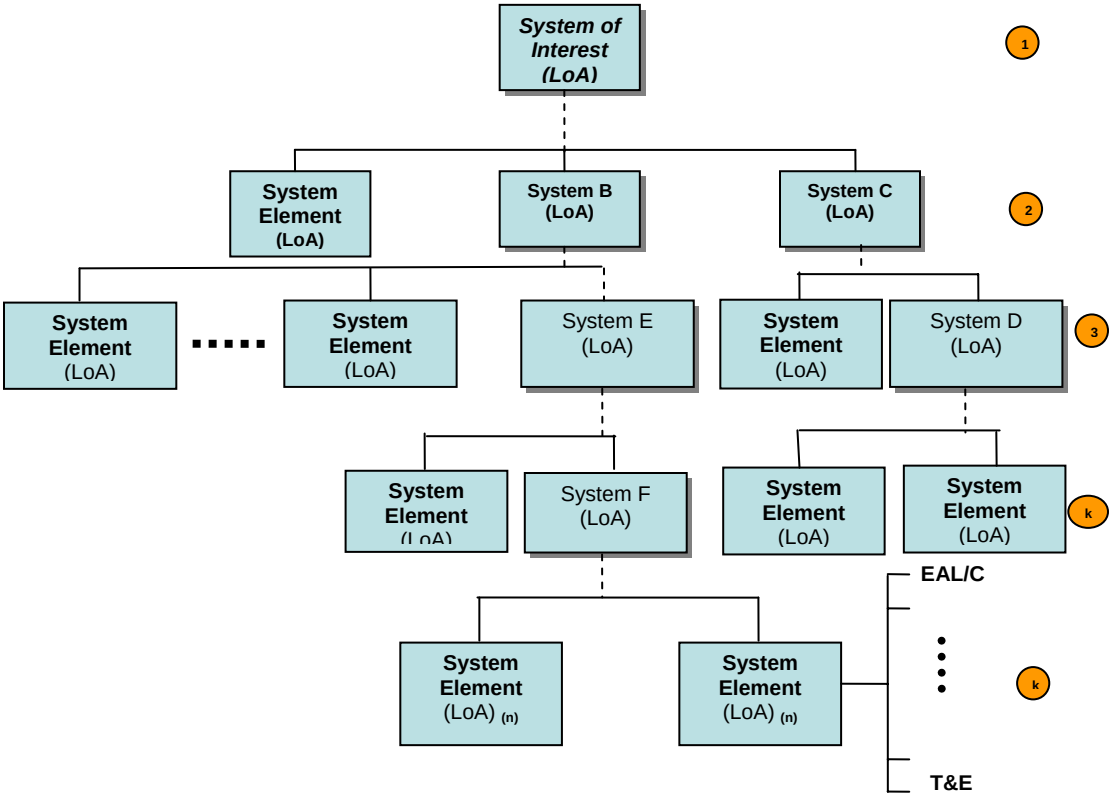
often be robust in the presence of attacks from portions of its environment, and these portions should be identified. The system may depend on properties of the environment, but these dependencies must be documented and reasonable.

3.4.2.2 Define required system functions

As the functions are defined, they must be further broken down into lower-level sub-functions (Figure 3-2), so that an analysis of alternatives can be performed (including build vs. buy decisions). These functions should include both:

1. Intrinsic functions, which directly deliver mission functions
2. Defensive functions, which secure a system and in turn deliver assurance

Some functions may be using COTS elements. In many cases the use of COTS elements should be encouraged, but if the COTS element can affect system assurance, the requirements on the COTS elements must be included in the system requirements. Issues to consider include additional functionality in the COTS that might negatively affect assurance, patching/upgrades (including whether the supplier controls this process), and default security settings. Building the system breakdown structure aids the SE by making explicit the specific functions, their inter-connectivity, associated criticalities of sub-functions, and the reasoning behind the criticality decisions. This breakdown structure provides a more manageable approach to building assurance into the system, thus assisting in providing the framework for developing the state of the system.



Note: Reference to Level of Assurance (LoA) of System of Interest, System and System Elements is to address notional view of a LoA. LoA for a System of Interest is a derivation of System and System Element level LoAs and the composability of those systems.

Figure 3-2 System Function and Level of Assurance Breakdown Structure

3.4.2.3 Identifying critical functions

A function is a critical function if it is (1) an intrinsic function that if compromised jeopardizes the mission, or (2) a defensive function that is protecting a critical function (which may be intrinsic or defensive).

The functional WBS should be augmented to factor criticality and the derived assurance criteria into the functional breakdown. This can be done by identifying the critical functions in the work breakdown structure.

Any assurance functions should be included in the Functional Baseline. The Functional Baseline should identify which functions have been determined to be critical.

Work to minimize the critical functions within a system in terms of number, size, and complexity. This minimizes cost, time to delivery/time to market, performance impact, and risk. By minimizing critical functions, the assurance case and traceability tend to be easier to manage. Note though that systems that are not known to be explicitly targeted should not be completely ignored as targeted systems may depend on non-targeted systems to accomplish their mission.

A single point of failure should be avoided if possible. Ideally, the system would implement layered levels of criticality and element redundancy, supporting graceful degradation while accomplishing the mission. Ideally, the failure of a single function might degrade system performance but not cause the loss of the system or the mission.

Define and extend the assurance case claims to identify derived system assurance requirements for all critical functions. Make sure that threats, intentional and unintentional vulnerabilities, and weaknesses are addressed.

3.4.2.4 Building the Assurance Case

The requirements definition and analysis process is where the assurance case begins to be addressed and where system claims are defined and identified as a subset of the system requirements. The functional breakdown of the system is performed, aiding in the identification of the critical functions in the system elements that need to be assured. The claims decomposition must be coordinated with the functional breakdown of the system, which leverages both the top level and technical requirements of the system.

- Engage stakeholders to establish the system top-level assurance claims under the constraints of budgets, time-to-market, implementation feasibility, as well as the ability to prove the claims. These claims should be directly linked to the assurance related the defined system requirements.
- Correlate the top-level claims with both the general and technical system requirements to review the accuracy of the claims.
- Decompose the top-level claims down into sub-claims and lower level sub-claims, and arguments. Consider whether adequate evidence is obtainable for the claims. Claims development is an iterative process similar requirement definition. And as such this effort may take several cycles.
- Bound the claims, sub-claims, and arguments in parallel with function criticality to make the assessment of threats, potential weaknesses, and associated vulnerabilities manageable.
- Link any assumptions made at each level of decomposition to sub-claims and arguments.

- Obtain stakeholder approval on the claims breakdown, assumptions, and the types of evidence to be collected.
- Restructure any claims that cannot be justified by collectable evidence.

3.4.3 Architectural Design

Architectural design is an important process that bridges identification of requirements to implementation. Unfortunately, this process is sometimes performed inadequately in a rush to production to meet time constraints. In addition, it is easy for a program to become dominated by its features or feature sets, especially in a time-critical operational environment, and non-functional requirements are sometimes shortchanged.

This section describes practices in architectural design that can improve assurance. The organization of this section is based on ISO/IEC 15288:2008 (Section 6.4.4), but some additional detail is extracted from IEEE 1220 (Section 6.3–6.7), which provides more detail in some areas.

3.4.3.1 Define appropriate logical architectural designs and methodologies

When identifying the architectural elements and their interactions, consider the following general issues:

- Attempt to separate elements that are highly critical from those that are not, and attempt to make the critical elements easier to assure by making them smaller, less complex, and more isolated from impact by other elements. Consider separating data, limiting the data and control flows, prioritizing the critical flows, and ensuring that such flows can be predictably and securely delivered (with confidentiality, integrity, availability, authentication, accountability (including non-repudiation), and auditability). Assurance efforts may need to focus on these critical “intrinsic” elements and flows.
- Separation and information flow are key. Separation mechanisms also include encryption, physical separation (e.g., different cables or machines), and mechanisms incorporated in the central processing unit.
- Include defensive elements (elements whose job is to protect elements from each other and/or the surrounding environment, or to monitor elements). This particularly includes defensive elements that examine inputs from potentially untrustworthy sources, so they can limit inputs to valid ranges and formats. Determine what defensive functions can be leveraged across multiple elements or the entire system. Assurance efforts may need to examine the defensive elements necessary to protect and monitor the critical elements.
- Identify elements and their interfaces to accommodate requirements, including not just functionality but also protection from abuse cases (as determined in the requirements).
- Individually secured elements are not enough; the composition of the elements and their interconnections also matter. Understanding the interrelationships between elements and their linkages will help in addressing potentially weak areas in the design.
- Use defense-in-depth measures where appropriate, i.e., use multiple independent layered mechanisms to protect a system’s critical functionality to make it more difficult for attackers to succeed.
- Beware of maximizing performance (including throughput) to the detriment of assurance. Consolidation often improves performance (compared to methods such as data/process separation), but users are not aided by high-performance subverted systems.

- Support some level of randomization and non-deterministic assignment of resources to help limit effectiveness of attacks that rely on the use of architectural knowledge (e.g., buffer overflow).
- Support identification and measurement of a trusted baseline so corruption can be identified.
- Support identification of corrupted and tainted resources and functionality, and recovery mechanisms to reestablish the system based on a trusted baseline.
- Collect additional information on assets, threats, and weaknesses of the architectures under consideration.
- Use the refined assurance case to derive additional requirements and design constraints.

Some specific approaches to consider when developing the architecture are:

- *Least privilege.* Elements should ideally be granted the least necessary privilege. Mechanisms for implementing least privilege include mutually suspicious elements (i.e., elements are designed to treat each other as untrusted or with limited trust), limited interfaces (instead of permitting broad direct access to a process or data, e.g., data in a database), limiting file system privileges, languages that enforce limited privileges, and so on. Access control mechanisms must be identified and be non-bypassable and tamper-proof. There are many mechanisms for addressing access controls depending on the development platforms used, including role- or attribute-based mechanisms. These in turn typically tie back into identity management.
- *Isolation/containment.* Isolation/containment mechanisms such as sandboxes, jails, layered interfaces, computer language security managers, element wrappers, containers, virtual machines, firewalls that are internal to the system, and so on can limit the damage that a less-trusted element might impose on the system. This may be especially important when integrating COTS or GOTS elements (in some cases GOTS may have a lower risk of subversion, but in such cases there is still a risk of unintentional vulnerabilities).
- *Monitoring and response for both legitimate and illegitimate actions.* Systems should detect both legitimate and illegitimate actions (including some faults), be able to record them, and potentially respond to the actions. A response may report, notify personnel (e.g., alarms), log in an audit record, block, hinder, or slow activity, perform recovery, or make other effective responses. This requires elements that can detect, record, and respond to such events and, typically, a communication path for their interconnection. Many of today's COTS products and platforms provide both audit and access control mechanisms that can be leveraged for both the system design and development. Where possible, consider using such existing mechanisms, in part because using them enables use of many industry tools (which link to these mechanisms) for monitoring the system in its operational environment.
- *Tolerance.* Systems should include tolerance for some bad events, including security violations. In many cases, it is better to provide graceful degradation or reduction in service to a minimal level, instead of a denial of service. Possible tolerance approaches include intrusion tolerance (including tolerance to partial attacker success), fault tolerance, robustness (input tolerance), damage isolation or confinement, continuation of service although possibly degraded, redundant operations/procedures (which may only provide reduced service), recovery of the system to a legitimate state (including "self-healing" approaches such as periodic restarts from safe states), and disaster recovery. An example of intrusion tolerance is a security gateway appliance that offers firewall, intrusion prevention, anti-virus, and intrusion detection services that operate as

independent services. A failure of one feature does not impact operation of the other security features. Additionally, if the hardware-based features like intrusion prevention fail, the system will revert to software intrusion prevention services. Isolation/containment mechanisms can be leveraged to aid tolerance. In addition, during deployment and maintenance, it is feasible to configure solutions to obtain both disaster recovery and continuation of service – reduced or full.

- *Identification and authentication mechanisms.* Identities should be defined and known for the entities involved; and their privileges or authorizations must be known or able to be established. This often requires some I&A mechanisms; more critical functions may require stronger mechanisms such as multi-factor authentication (e.g., a smartcard plus password) or integrating into a public key infrastructure (PKI) instead of simple password mechanisms. Operational environments may already provide some I&A mechanisms; normally in such cases they should be leveraged, but ensure that they provide the level of strength necessary to meet the need for system criticality or determine how such mechanisms can be strengthened.
- *Cryptography.* Cryptographic functionality, where used, must be FIPS PUB 140-2 compliant. Cryptography is a foundational security element that is often needed to implement I&A, confidentiality, integrity, availability, authentication, accountability (including non-repudiation), and auditability. Although they have been widely used in the past, some cryptographic algorithms (such as Data Encryption Standard (DES)) and protocols (such as Wireless Equivalency Privacy (WEP)) are inadequate for today's needs. Consult with security experts when applying cryptography, since it can be easily misapplied, by architects and developers, without detection by system developers.
- *Deception.* While it provides no guarantees, deception may be useful for aiding concealment, intelligence gathering, confusing the attacker, and wasting an attacker's resources. Use this type of resource to protect a system against intentional or unintentional attacks during deployment and maintenance. Examples include fake elements (e.g., honeypots and honeynets), falsified data, misleading service name/version number reports, and nonstandard ports.
- *Employ interface standards or standard elements*. In some areas there are standards for element interfaces and/or widely used elements. Using interface standards and/or standard elements for which an assurance case has been established often makes it easier to gain assurance, since typically the work to gain assurance can build on others' work. In addition, using interface standards may ease switching from one implementation to another if the first implementation has serious assurance problems.

Anti-Tamper Techniques

Anti-tamper techniques should be considered. Anti-tamper mechanisms can be divided into tamper-evident, tamper-resistant, and tamper-respondent mechanisms; systems may have multiple mechanisms. Examples of anti-tamper mechanisms include:

1. Tamper-evident (break-once) seals
2. Hard-to-forge images (e.g., holographs)
3. Protected enclosures (e.g., closed via epoxies, one-way screws, etc.)
4. Binding epoxy that encapsulates key elements (e.g., critical integrated circuits)
5. Protective packaging (coatings such as laminates)
6. Algorithms adjusted to have constant power use (to minimize emanations), countering power analysis techniques (including simple and differential power analysis)

7. Soft architectures – i.e., building generic hardware and/or software, and as late as possible specializing the system with specific software or configuration data and/or techniques. Examples include use of FPGAs, reconfigurable computing, and separately protected software plug-ins. This reduces the opportunity for tampering or unauthorized disclosure.
8. Tamper-respondent enclosures (e.g., if someone drills into it, zero all data)
9. Use of multiple on-board monitors to determine if results are as expected, and respond (such as disabling device, removing sensitive data, and/or report incident) if results differ
10. Use of multiple mechanisms as part of a defense-in-depth approach

All of the above are merely examples; this is not a complete list. FIPS 140 provides some additional information. (At the time of this writing, there is a draft FIPS 140-3 that adds more information on algorithms with constant power use.) The Web site http://sidechannelattacks.com also has relevant information.

Other anti-tamper techniques include trusted processing, physical tamper detection mechanisms, the use of intrinsically tamper-resistant hardware elements/architectures, physical disabling/removing of access lines/ports, and encryption key hiding/management.
Some anti-tamper techniques can limit testability (such as time to perform diagnostics and ease of maintenance), maintainability (such as time to verify and validate fault and repair tasks/actions), or reliability (false alarms caused by anti-tamper checking). Consider trade-offs as well as process changes such as when or how to implement the anti-tamper techniques. Ensure that protected elements are not subverted before being protected by anti-tamper mechanisms, since such mechanisms tend to hide information that would expose such subversion. Also, ensure that the anti-tamper mechanism cannot be exploited by an adversary (i.e., an adversary should not be able to trigger a denial-of-service attack through the anti-tamper mechanism).

3.4.3.2 Partition and allocate identified system functions

ISO/IEC 15288:2008 reminds PMs and SEs to "Generate derived requirements as needed for the allocations." When allocating requirements to architecture elements, it may become immediately obvious that certain elements/elements are much more critical than others; if they can be isolated from less-critical elements, assurance is easier to obtain.

Derived requirements often imply a verification approach, and may include measures of performance (MOPs) associated with assurance, such as:

- Use of certified elements (obtaining certification data as evidence), such as Common Criteria or FIPS 140-2.
- Static analysis tool results (when analyzing source or binary)
- Security or defensive functionality testing results
- Fuzz testing requirements, possibly with specification of valid/invalid input ranges
- Define acceptable input, as minimally as possible, and require the element to reject all input that fails to be acceptable. Avoid defining "unacceptable" input, even though this is logically equivalent, because attackers can often create malicious inputs that do not match a pattern of unacceptable values.
- Encryption for both data at rest and in motion, including encryption strength

3.4.3.3 Analyze the design to establish element criteria

This analysis process must consider assurance, to determine the necessary criteria for each element and how the elements must be combined. The analysis must ensure that the architecture can meet assurance requirements, including a determination that it addresses all identified abuse cases. Use system assurance requirements, design constraints, and system assurance critical scenarios for architectural trade-off analysis and document the results. Include the following in the architectural analysis:

- Identify and evaluate architecture vulnerabilities.
- Allocate system assurance requirements to architectural elements.
  - Consider defensive architectural elements that allow an assurance requirement to be allocated solely to that architectural element.
  - Examine intrinsic and defensive functions.
- Perform Failure Mode Effects and Criticality Analysis (FMECA), fault tree analysis (FTA), anti-tamper analysis, and other applicable techniques to evaluate the architecture system assurance properties.
- For interfaces evaluate the filtering restrictions that the elements must enforce. For example, for data interfaces, identify the legal values (so that all illegal values will be rejected/trapped); for electrical connections, identify where voltage limits must be enforced.
- Update and refine the system assurance case with claims, arguments, and expected evidence for the selected architecture.
- Protect architectural information from potential adversaries by preventing public access to it.

As issues are identified, designs may be modified, and/or the identified issues are input to the risk management process (so steps may be taken to prevent, reduce the impact of, or accept the risks). Often, specific elements will be more critical, and these should be identified so that special efforts can be made to assure them, as discussed notionally in the requirements section of this document. Care must also be taken that critical protective elements, for example, authentication, are not bypassed and that needed separations among system elements are preserved. Some existing documents identify criticality levels that may be useful for identifying element design criteria (e.g., from catastrophic to no effect).

A variety of analysis techniques exist, depending on factors such as the type of system, technologies used, and issues being analyzed. These include:

- Threat analysis, including:
  - Threat analysis involves identifying a threat, modeling a set of possible attacks, assessing probability, potential harm of the attack, and trying to minimize or eradicate the threats
- System analysis, including:
  - Failure Modes and Effects Analysis (FMEA)):, but focused on assurance issues, e.g., both unintentional vulnerabilities and element subversion: http://www.fmeainfocentre.com/
  - Failure Mode Effects and Criticality Analysis (FMECA): http://www.fmeainfocentre.com/presentations/fmeca.pdf

- Failure Modes and Impacts Criticality Analysis (FMICA): http://vva.dmso.mil/Special_topics/Risk/Risk.htm
- Fault Tree Analysis (FTA): http://reliability.sandia.gov/Reliability/Fault_Tree_Analysis/fault_tree_analysis.html
- Vulnerability trees: http://www.glam.ac.uk/socschool/research/publications/technical/CS-03-2.pdf

Architectures can be examined to determine if the principles of least privilege, limited data access, redundancy where appropriate, heterogeneity where appropriate, and so on, have been successfully applied. Where appropriate, consider adding "test interfaces" (to allow insertion or extraction of test data) and/or built-in tests, but ensure that these interfaces are properly protected so that they are not exploitable. Develop tests (including those for assurance) before implementation to increase the likelihood of correct results. Consider the design's extensibility, particularly to determine the difficulty of supporting probable future changes, and to reduce the difficulty of maintaining assurance in those changes.

Critical elements may need to be hardened or have their criticality managed. Techniques such as graceful degradation, isolation, reducing single points of failure/multi-pathing, modularity, diversity, and use of interchangeable standards-compliant elements should be encouraged to significantly reduce the number, size, and/or impact of critical elements. Residual risks inherent in the use of less-assured element products can then be better managed. Isolation techniques can be applied at many levels, and the isolation mechanisms themselves need to be assured so that they cannot be easily subverted.

A result of this analysis is an argument that demonstrates that the assurance claims (requirements) have been met, and what evidence will demonstrate that the argument is valid (e.g., the framework of an assurance case). These design criteria are allocated to the appropriate system elements.

Design criteria may take other forms (e.g., requiring the use of static analysis tools on source code and/or binaries, fuzz testing, formal proofs of design and/or code, peer reviews, taint checking. See Section 3.4.3.2). Such criteria eventually link to the system verification process.

3.4.3.4 Determine which system requirements are allocated to operators

"Operators" include administrators as well as ordinary users. From an assurance perspective, operators must be granted the privileges they need to perform their tasks. In instances where these privileges are critical, information must be secure. Consider limiting the access of operators (possibly by separating their roles), and/or logging their activities, since not all people are trustworthy. Constantly asking users for permission to perform tasks should be avoided; users may not have the information to reliably make that determination, and over time may choose to automatically accept risks without thinking. See Section 3.4.3.1 regarding access control, including role-based and attribute-based access control.

3.4.3.5 Determine whether hardware and software are available off-the-shelf

As noted in ISO/IEC 15288:2008, these elements must satisfy the design and interface criteria. Off-the-shelf elements should be considered for an eventual make/buy decision (for U.S. government acquisition this is required). In addition to their functional requirements, however,

there is a need to perform a supply chain analysis and for software, perform source code review(s), depending on the criticality of elements, to sufficiently assure them. (See Section 3.1.2, Supply Process).

As noted earlier, where interface standards exist, strongly consider using such standards. Standards tend to be examined and vetted through large groups, reducing certain kinds of risk. Where practical, limit element use to such standards so that elements can be replaced later with another standards-compliant element.

3.4.3.6 Evaluate alternative design solutions

ISO/IEC 15288:2008 notes that these alternative design solutions should be modeled "to a level of detail that permits comparison against the specifications expressed in the system requirements and the performance, costs, time scales (including time to market/deployment), and risks expressed in the stakeholder requirements."

It is a good idea to identify and review various architecture alternatives. Examining the set of critical elements (to potentially reduce their size or number) may be especially helpful in identifying alternatives. Architectural alternatives include "build vs. buy," and leveraging partial or full solutions. In all cases, consider the effect on assurance of these alternatives, including whatever mitigation techniques may be necessary.

3.4.3.7 Define and document the interfaces

ISO/IEC 15288:2008 notes that these interfaces are both "between system elements and at the system boundary with external systems." This documentation should include information on the assumptions necessary for assurance.

Identify opportunities for new standards, and include assurance considerations in those new standards. One challenge to using external elements is to know their relevant properties, so standards for such properties can be helpful. These include product-level assurance properties, expressing reference models/standards/requirements in modeling language, system interdependencies with the supporting infrastructure, susceptibility to external environment (e.g., EMI), and identifying methods for validating compliance with requirements/standards, using industry-developed tools. For example, it would be good to have standard isolation models, so element developers do not depend on access(es) they are unlikely to receive in real-time system operation.

3.4.3.8 Specify the selected physical design solution

ISO/IEC 15288:2008 notes that the physical design solution should be specified "as an architectural design baseline in terms of its functions, performance, behavior, interfaces and unavoidable implementation constraints."

3.4.3.9 Maintain mutual traceability between architectural design and system requirements

This traceability must be a living document, including traceability for assurance requirements, so assurance can be maintained through changes in requirements or implementation.

3.4.3.10 Building the assurance case

While performing architectural design, update the assurance case to include the necessary arguments (which justify why the architectural design will support meeting the assurance claims):

- The architectural design should identify the interfaces to untrusted sources; justify in the assurance arguments that this is the complete set of such interfaces. For these interfaces:
  - Identify the full set of elements that connect to or implement these interfaces.
  - Justify why the measures taken to harden these elements from attack are adequate, and
  - justify that whatever input or output filtering the elements will perform is sufficient.
- Some assurance arguments may be based on a defense-in-depth approach. In these cases, identify the primary hardening methods and explain why (1) an attacker would have to defeat all of the methods to succeed (instead of a subset) and (2) why this would be much more difficult than defeating just one method.
- Some assurance arguments employ element hardening, which often includes creating and enforcing least-privilege limits on elements connected to external interfaces. In these cases, explain why these limitations would significantly reduce the effectiveness of an attack or successful penetration.
- Externally-connected elements often cannot completely filter all possible malicious input. In this case, the assurance argument should demonstrate that the other elements that process the input will appropriately respond to malicious input (this is particularly a challenge for multimedia formats).
- Develop the assurance case, including its arguments, in a way that makes it easy to change. Use tools to maintain the assurance case, and divide up the assurance case into smaller modules so changes can be localized.

3.4.4 Implementation

During the implementation process (ISO/IEC 15288:2008 Section 6.4.5), the system capabilities, interfaces, and constraints are transformed into system elements (such as fabricated hardware (HW) and coded software (SW)) that satisfy requirements and the architectural design. It is a demonstrated fact that even a well-designed system can fail if its key elements are implemented poorly, which underscores the criticality of system assurance considerations during implementation.

No single assurance technique can prevent vulnerabilities during implementation. Therefore, various techniques are necessary, each of which has some probability for detecting and avoiding various vulnerabilities that would otherwise be inserted during implementation. Selected detection techniques should be sufficiently different so that they are not correlated in their likelihood of addressing a given type of vulnerability. Performing multiple techniques identified herein will decrease the overall probability of vulnerability insertion during implementation.

3.4.4.1 Implementation strategy (and integration strategy)

It is important to follow documented strategies for both implementation and integration. While the Implementation Process (Section 3.4.4) and the Integration Process (Section 3.4.5) are treated separately, the strategies required for both have many overlapping considerations for system assurance. Consequently, they are both treated in this subsection.

The Implementation Strategy and the Integration Strategy can either be part of the updated project plan or be generated as separate documents prior to implementation and integration. Specific considerations for system assurance should be included:

- The balance between assurance and system performance (carefully considering both).
  - Avoid high performance at the cost of subversion.
- Include mechanisms for revelation, isolation, and diagnosis of faults.
  - Implement elements and integrate the system for testability.
- Consider implications of fault-tolerant techniques that may disguise important warnings (e.g., of attack) or create vulnerabilities (e.g., if some elements ignore bad data but others do not, creating an inconsistent state).
- Assurance of custom-designed HW and SW for implementation/integration
  - Secure the development and integration environments and tools by appropriate isolation (physical security and access measures), even when the environments are Unclassified.
  - Include assurance-specific checklists into the SW peer review process.
  - Emplace tools to identify unintentional/intentional vulnerabilities.
    - (Identify expected outcome of tool usage, and what to do with it)
- Assurance of reuse HW and SW, including COTS and GOTS
  - Concentrate on the strategy for SOUP, as its ability for adaptation to adequately assured reuse may constrain the overall design.
  - Adaptation of reuse SW conducted with regard to applicable security and privacy standards.
  - Emplace tools to identify unintentional/intentional vulnerabilities.
    - Identify expected outcome of tool usage, and what to do with it.
- How system elements received from suppliers and/or retrieved from storage will be checked/certified for assurance purposes (e.g., they might be verified against acceptance criteria specified in an assurance agreement).
- Clarification of assurance-related issues to be addressed during the SETRs (Systems Engineering Technical Reviews) occurring during implementation and integration. Examples of the relevant SETRs might include:
  - Systems Requirements Review (SRR)
  - Systems Functional Review (SFR)
  - Preliminary Design Review (PDR) (down to the Configuration Item (CI) level)
  - Critical Design Review (CDR) (down to the element and unit level)
  - Test Readiness Review (TRR) (down to the lowest level of algorithms and boards)
- Protection of system boundaries and interfaces.
- Protection of the development and integration labs.
  - Establish a routine examination of all tools introduced into the labs.
  - Prevent “dumpster diving” by monitoring trash for unintentional “releases” (destroy trash).
- Appropriate system assurance training for staff
  - Provide counterintelligence briefings for staff who will interface with outsiders, as part of the supplier assurance effort.

    - Provide adequate training on assurance techniques and tools for effective application (e.g., difficulties for implementation and integration arising from maliciousness).
- Adequate CM-related access control and tracking measures
    - Limit privileges to authorized personnel.
    - Require two-person integrity, as needed.
    - Implement repository subversion countermeasures.
    - Configure HW and SW elements with regard to applicable security and privacy standards.

Tools are a key enabler for assurance. Although they are not a panacea, they provide both insight and an approach to reducing the risk of vulnerabilities. Industry provides some tools for the identification and assessment of unintentional as well as already known intentional vulnerabilities (e.g., viruses). Tools and techniques for detecting not-yet-identified malicious code (for example, by analyzing its behavior) still have limitations although continued efforts are being made to improve heuristics. In some cases tools for proving properties of designs or program may be appropriate. Organizations should evaluate, as early as practical, the various options to determine if such tools are adequate for their specific purposes, and include this analysis in their strategy.

3.4.4.2 Constraints imposed by the strategies on the design

It is important to prioritize and handle the unavoidable constraints of implementation and integration that influence any system assurance requirements.

Difficulties for implementation and integration arising from maliciousness should be covered within the Implementation/Integration Strategy, including their effects in (potentially) constraining the design. For example, element and system realization (both SW and HW implementation/integration) can include malicious logic that causes:

- Loss of confidentiality
- Loss of integrity (e.g., changing information into invalid results or producing the wrong results)
- Loss of system availability (a critical vulnerability can cause entire system/mission failure)
- Possibility of repudiation

The adaptation of reuse SW must be conducted with regard to applicable security and privacy standards. This may lead to constraints on the system design. In fact, the strategy for adequate assurance of reused HW and SW elements in general (COTS, GOTS, and especially SOUP), may imply limitations in potential reuse, which in turn may constrain the overall system design.

Protection of system boundaries and interfaces may also lead to design constraints. Specific techniques should be employed to verify that interface breaches are prevented through adequate design and access controls (see Architectural Design).

3.4.4.3 Countering threats in implementation

Malicious SW or HW is difficult to counter because it is hard to detect. While code is being developed and HW elements are being fabricated, the following things should be kept in mind by the Lead/Chief Systems Engineer in guiding and overseeing the development team:

- Vulnerable elements can be extremely small relative to the size of the overall system. Today's software and hardware systems are often large and complex, with many interactions. Yet it may take only a few lines of malicious code or hardware logic to cause great damage, and these elements can be distributed across the system, making vulnerabilities easy to hide and difficult to find.
- Effects are typically nonlinear. It is important to remember (especially with digital systems) that a slight change in input can yield massive effects.
- Malicious implementation can appear valid. Publicly known examples have shown that it is easy to create code that is malicious yet at first glance appears to be legitimate.
- Elements may lack transparency. Many commercial elements are delivered in forms that are difficult to analyze (e.g., binary-only forms for software and IP cores for hardware). For external services (e.g., software-as-a-service), there may not even be a binary that can be examined. This makes attacks easy to hide.
- Malicious implementation can pass tests but then fail when most critical. An intelligent adversary can develop attacks that will occur when the adversary desires—when it is the worst possible moment for the system. This requires considerations that are fundamentally different from typical safety analyses (which depend on probabilistic behavior).

The implementation process should incorporate frequent threat updates to ensure that countermeasures are developed based on the intent and specific threat agent capabilities that may change throughout the system life cycle. For different kinds of threats, different countermeasures and avoidance techniques apply, for example:

- **Threat: Unintentional vulnerability inserted during development**

  Countermeasure/Avoidance Techniques:

  - Prefer development tools/languages that prevent or warn of common vulnerability and error types. All programming and hardware description languages have constructs that are undefined, imperfectly defined, implementation-dependent, or difficult to use correctly. Both unintentional and intentional vulnerabilities may result (intentional vulnerabilities can masquerade as unintentional errors). The language should aid the design and development of reliable systems. The language should be designed to avoid error-prone features and to maximize automatic detection of implementation errors as soon as practicable. Risky properties of languages include failures to protect against boundary overflows and null pointers, or constructs that are easily misinterpreted. Some common constructs of C and C++ are prone to misuse, making the risk of including vulnerabilities especially likely in these languages. When using languages that are especially risk-prone, use additional techniques to reduce their risks. These include other techniques identified in this guidebook (e.g., static analysis tools, peer reviews/inspections, additional training, and specially skilled personnel). In some cases, a secure subset of a language may be appropriate. If switching to a language unfamiliar to the developers, include the necessary training for that language. It may be more cost-effective to retrain existing developers in a more-secure language than to attempt to implement additional necessary measures in

a less-secure language. There is existing international standards work to develop guidance for avoiding vulnerabilities through language selection and use; if available, consider it. Before selecting tools/languages, be sure to include these additional measures in the total costs, since in some cases these additional measures may exceed the "savings" through using less-secure languages.

- Enable HW/SW compiler "warnings" as much as practical. These warnings should be enabled as soon as possible, especially for custom code, because "warnings" are often hard to address after implementation begins (due to the potentially overwhelming number of warnings to be addressed). Since compilers add new warnings over time, implementers may need to reevaluate which warnings are appropriate to include or add during maintenance.
- Train developers to recognize common mistakes that lead to vulnerabilities (e.g., buffer overflows, cross-site scripting, and integer overflows. Sources of such lists include CWE and the Open Web Application Security Project (OWASP) top ten.)
- Gracefully handle out-of-bounds data (SW) and signals (HW), e.g., reject such inputs and present an indicator that input has been rejected. Out-of-bounds data and signals should be logged for the purposes of detecting trends (such as attack indicators).
- Use tool-enforced HW and SW implementation guides to help avoid common mistakes and anomalies.
- Use static and dynamic tools to identify vulnerabilities. Vulnerability identification tools generate false positives and false negatives; therefore, use a suite of tools and emphasize eliminating the riskiest constructs.
- Select development tools with proven track records of low defect-insertion rates. (Some development tools can insert hard-to-find defects.)
- Perform peer reviews of implemented custom elements that search for vulnerabilities. (It may be useful to include security engineering in the peer review.)
- Perform periodic testing of elements for functionality and reliability requirements. Some additional test such as fault insertion may be key to detect malicious SW or HW threats.

➢ **Threat**: **Intentional vulnerability inserted by a solo authorized developer**

Countermeasure/Avoidance Techniques:

- Use CM tools to restrict access to software elements based on "need to know" and audit access.
- Use development languages that enforce limited access. (Programming languages like Java have mechanisms such as "private" and Security Manager. In contrast, C++'s version of "private" is easily subverted by a malicious authorized developer through mechanisms such as pointer manipulation.)
- Use tool-enforced HW and SW implementation guides. (Tool-enforced implementation guides can prevent some hiding techniques.)
- Use tools to search for already-known as well as not-yet-identified malicious code. (Tools and techniques for detecting not-yet-identified malicious code (for example, by analyzing its behavior) are not mature.)
- Integrate advanced security testing techniques that can be used to identify zero-day vulnerabilities as well as embedded malicious code. Advanced security testing techniques may include fuzzing and automated source code analysis that can reduce

vulnerabilities and deter adversaries by introducing the possibility of discovering covertly inserted code.

- Use paired development and perform peer reviews of implemented elements.
- Ensure that what is reviewed is the same as what is used (see CM section).
- For hardware fabrication, limit the access and controls provided to operators and/or include monitors (such as cameras) to increase detection of subversion.
- Implement frequent audit controls of processes/procedures.
- Test sampled hardware parts for critical test parameters.

➢ **Threat: Intentional vulnerability inserted during development by a solo unauthorized person**

Countermeasure/Avoidance Techniques:

- Enforce CM processes during development (see CM section).
- Ensure that the SW reviewed is the same as what is used (see CM section).
- Ensure limited access during hardware fabrication.
- For mass production, include CM procedures for difficult-to-forge markings on hardware elements (so that it will be difficult to swap them later).
- Use development languages and tools that enforce limited access.

➢ **Threat: Intentional vulnerability inserted during development by a supplier or supplier's supplier.**

Countermeasure/Avoidance Techniques:

- Use tools to search for already-known malicious code (e.g., viruses). (Tools to search for not-yet-known malicious code are extremely immature.)
- Perform additional checks and defensive measures at the implementation level for data and signals entering and leaving the element.
- The implementation process has difficulty countering this threat. Therefore, it is primarily countered by other system life cycle processes, such as architectural design, as well as supply.
  - ♦ Architecture and design will identify such techniques as sandboxing to reduce impact of malicious elements through containment.
  - ♦ Supply management processes will reduce the risk of malevolent products.

The above examples of threats and countermeasures are a sampling of the many possibilities and are not intended to be a comprehensive list.

Many implementation processes include implementation guides, which may also be termed "coding standards" or "style guides." For assurance, these need to include information that helps counter vulnerabilities (e.g., counter the use of insecure constructs), and not be merely formatting guides. The guide should identify the subset of the language that is acceptable for use, and it should include the rationale for that subset.

Ensure that any assessment or evaluation of an OTS element is relevant to its intended operation in the system before depending on that assessment or evaluation.

Implementation issues are encountered cyclically within incremental and spiral development efforts and within the execution of system/element Engineering Change Requests (ECRs) in

general. The level of rigor needed, vulnerability, and closure hazards of ECR implementation should be addressed by assurance engineering, and integrated into the systems engineering tasks of ECR analysis, design, implementation, integration, and testing. Trouble Reports and the follow-on Change Requests should be monitored for evidence of problems in assurance-critical functions.

Systems engineering personnel also get involved during the implementation process with assurance-related requirements that need to be addressed during the late life cycle V&V processes. The SE should verify that a test procedure exists to test each assurance-critical requirement, function, and signal or message. The SE should also develop regression analysis and test planning to ensure that a given element in new development/modification cannot adversely affect another element.

3.4.4.4 Record evidence that system elements meet assurance requirements

Document the successful execution of the assurance-related elements of the Implementation Strategy. In particular, record the assurance claims that can be made with respect to the implemented system elements, the arguments that justify those claims, and the evidence that supports the arguments.

Record any non-conformances introduced or discovered during implementation, and initiate action plans to resolve those issues that affect assurance.

3.4.4.5 Package and store system elements appropriately

Once implemented, it is important to package and store the system elements in accordance with prior assurance agreements. Part of the task of protecting system boundaries and interfaces includes packaging and storing the implemented system elements, prior to their transition to operations, by containment and conveyance methods that achieve the required level of assurance.

3.4.4.6 Building the assurance case

While performing implementation and integration[1], update the assurance case to complete the arguments as well as to add some of the evidence needed to justify the assurance claims:

- Complete the claims, sub-claims and arguments with a final verification.
- Begin collecting the evidence on the interfaces to trusted and untrusted sources, including evidence of implementation of defensive functions and secure interfaces. For interfaces to untrusted sources, identify the hardening and/or filtering approaches used.
- For defense-in-depth approaches, collect arguments and evidence. Evidence collected should support the justifications (arguments) such as the traceability to defensive functions.
- Adjust the arguments and evidence as appropriate where there is a changing (dynamic) enterprise environment. This is critical when leveraging existing enterprise infrastructure functions (e.g., network infrastructure, encryption, or identity and access management), to confirm that the infrastructure continues to support the assurance case.

[1] Many of the issues in building an assurance case are common to both the implementation and integration processes, so this document has a single section that covers both processes.

- Collect evidence from static analysis. Such evidence may include results from inspecting for weaknesses, such as buffer overruns or not validating inputs.
- Review claims and arguments for feasibility. In some cases, it may be determined that some evidence is not collectable, triggering a need to update the claims and arguments.

3.4.5 Integration

During the integration process (ISO/IEC 15288:2008 Section 6.4.6), a system capable of being verified against the specified requirements of architectural design is assembled and integrated. Hardware, software, firmware, and other system elements are combined to form complete or partial system configurations that create a product specified in the system requirements. It is essential during integration to ensure assurance of both elements and their interfaces. In today's environment, this integration will often also need to consider the integration of systems of systems.

It is arguable whether today's systems, and systems of systems, are more developed or integrated. As systems become more complex, their elements should be more modular, better defined, and more easily integrated, in order to ensure the tractability of the assurance task. In the interim, the integration process requires even more careful attention to system assurance.

Since a system is susceptible to threats posed by vulnerabilities and malicious actions during integration, a strong integration strategy should be defined and followed. Some elements may present additional risks; those risks should be assessed in terms of their impact on the system (or SoS), and mitigated (e.g., through isolation and containment mechanisms) during integration. System elements should be integrated with solid interface control descriptions and detailed assembly procedures, using clearly specified integration facilities. Further considerations on this defined strategy are given in the next section.

3.4.5.1 Integration strategy

It is important to follow documented strategies for both implementation and integration. While the implementation process (the previous major subsection of this guidebook) and the integration process (the current major subsection of this guidebook) are treated separately, the strategies required for both have many overlapping considerations for system assurance. Consequently, they are both treated in the Implementation section. (For details, see "Implementation Strategy" and "Constraints Imposed by the Strategies on the Design" in the Implementation section.)

3.4.5.2 Obtain system elements

If a supplier has not provided sufficient information to confirm the operation of its element and the ability to assure it, then the assurance case for the integrated system may be more uncertain. More of this concept is discussed in the Supplier Assurance section.

3.4.5.3 Ensure that system elements have been verified

Ensure that any assessment or evaluation of an OTS element is relevant to its intended operation in the system before depending on that assessment or evaluation.

3.4.5.4 Integrate system elements

For different kinds of threats, different countermeasures and avoidance techniques apply, as follows:

- **Threat: Malicious substitution of elements and the unintentional use of incorrect elements during integration**

  Countermeasure/Avoidance Techniques:

  - Ensure that assembly of the system is consistent with the architectural design.
  - Perform an integrity analysis (and corresponding C&A) of integrated products.
  - Evaluate COTS/GOTS product vulnerabilities.
  - Assure the supply chain by evaluating suppliers' assurance cases. Do not take the state of supplied elements at face value; rather, ensure that the system is not threatened by the state of its integrated elements.
  - Identify and analyze public domain software to be integrated (including modifications to previously approved products).
  - Assure integrity of SW versions and HW items through adequately robust CM tools.
  - For mass production, include CM procedures for difficult-to-forge markings on hardware elements (so that it is difficult to swap them).
  - Perform tests as required to ensure the functionality and reliability of critical parts. Some additional test such as fault insertion may be key to detect malicious SW or HW threats.

- **Threat: Malicious or unintentionally incorrect system composition**

  Countermeasure/Avoidance Techniques:

  - Ensure that assembly of the system is consistent with the architectural design.
  - In integration facilities, limit the access and controls (possibly including monitors, such as cameras, to increase detection of subversion).
  - Revisit the "least privilege" criteria and test at a system level.
  - Ensure (perhaps through adequate architecture) that the security model is correctly implemented and that received transactions are expected and validated.
  - Ensure clear and detailed build-and-release procedures (including checks and balances) through adequately robust CM tools.

    Detailed Consideration: At each level of integration including the system level, it will become clear whether the various elements developed and bought off the shelf can be configured in the same pattern that the architecture specifies. Systems often enter integration as incomplete conglomerations of code and parts that require finishing activities. Integration engineers may generate their own concepts of how to perform this finishing job, which in turn may weaken the assurance case (because there is no documented trail from architectural decisions through design and implementation to integration).
  - Document all integration decisions (that finish the design-to-architecture fit) and relate them back to previous decisions. Update and evaluate the architecture, if necessary.

- Include integration engineers in the systems engineering process to affirm and reinforce the assurance case and to ensure configuration and integration information is well-documented and evaluated.
- Use the CM system to document and track architectural changes made to finish the integration (to support evaluating the assurance case).

➢ **Threat: Malicious or unintentionally incorrect interfaces**

Countermeasure/Avoidance Techniques:

- Assure that assembly of the system is consistent with the architectural design.
- Test all interfaces between elements.
- Perform network security testing and accreditation.

  Detailed Consideration: An interface is rarely as simple anymore as one module calling another. Integration touch points exist throughout the system, between software and hardware, among programmable logic devices, and among major subsystems including databases, security systems, transaction monitors, and operating systems.
- Check each touch point during integration. The correct functioning of intermediate and final interfaces cannot be assumed (e.g., the functions of a well-defended element may be completely undone by a database with significant exposures).

  Detailed Consideration: Interface control descriptions must be comprehensive (including knowledge of element interoperability and data transfer protocols) in order to avoid such problems as can be caused by buffer overflows. Also, for multi-layer stacks (a typical communications and networking architecture pattern), good architecture will ensure that the layers underneath operate as they should; but in a "trust, but verify" posture, it is important for layers above not to over-rely on this security.
- Integration engineers must verify all interface controls (which includes checking for the vulnerabilities of buffer overflows).
- Integration engineers should validate the specifications and agreements that document the data transfer in a multi-layer stack, including both explicit and implicit interfaces throughout the stack. In many cases, upper layers will need to validate their layer's data, since lower layers will not have the necessary information to do so.
- Examine or develop a security agreement when one sub-system exchanges sensitive information with another sub-system at a different location, particularly if it has different organizational controls. One example is an Interconnection Security Agreement (ISA). According to NIST, an ISA is established between the organizations that own and operate connected information systems to document the technical requirements of the interconnection. The ISA is a security document that specifies the requirements for connecting the information systems, describes the security controls that will be used to protect the systems and data, and contains a topographical drawing of the interconnection. It is a commitment between the owners of two systems to abide by specific rules of behavior. These rules are discretionary and should be based on risk. See NIST 800-47, "Security Guide for Interconnecting Information Technology Systems."

3.4.5.5 Record integration information

Document the successful execution of the assurance-related elements of the Integration Strategy. In particular, record how security properties and behaviors have been maintained. More importantly, record any non-conformances due to integration activities, and initiate action plans to resolve those issues that affect assurance.

Document any further justifications for system assurance that have been identified during integration, together with supporting evidence.

Changes made during integration may affect assurance, so they need to be documented and related back to previous decisions, to update the architecture if necessary, and evaluate the assurance implications of that change.

3.4.5.6 Building the assurance case

See "Building the Assurance Case" text in the Implementation section (Section 3.4.4.6).

3.4.6 Verification

The verification process (ISO/IEC 15288:2008, Section 6.4.7) is needed throughout the development of the system to ensure that the design solution has met the system requirements and to provide assurance that the system is being built "right." A "verified" system is able to demonstrate that it has been implemented correctly and is compliant with stakeholder system requirements, as documented. Requirements should be traced to implementation and verification using a requirements verification matrix. The system assurance requirements should be tagged with attributes assigning them to/as SA. In general, this implies a need to conduct some form of assessment, such as developmental testing and evaluation.

The verification process activities must be conducted to provide objective evidence as the basis for the assurance arguments. For additional information on verification approaches, see ISO/IEC 15408, NIST 800-53 and 53A, DoDD 8500.01E.

3.4.6.1 Define verification strategy and plan

The Verification Strategy and Plan must be enhanced to address any additional tools and techniques specifically needed to verify system assurance. Some verification techniques for system assurance may require additional element information than would otherwise be available; for example, when integrating COTS, access to source code may be necessary. The Verification Strategy and Plan generic approaches may be modified to incorporate consideration of system assurance. Depth and breadth of verification should be commensurate with risk.

In particular, ensure that the strategy and plan for assurance verification can be executed by individuals with the appropriate skill sets for the assurance tools and techniques. Such skill sets, which are specialized and less commonly available, include understanding common weaknesses and security engineering practices, including more specific skill sets such as analysis of assurance tool output to determine their meaning and associated impacts, ability to identify probable attack scenarios, reverse engineering techniques, and malicious code analysis.

3.4.6.2 Identify and communicate constraints on design decisions

Highly critical elements imply the need to strongly limit their size and intrinsic complexity, so that thorough verification can take place with those elements. The availability of assurance tools and techniques that can be applied may significantly impact the design. For example, many static analysis tools only focus on a few weaknesses and do not cover many others (which would need to be countered in other ways); some development tools/languages facilitate vulnerabilities that are difficult to avoid and for verification tools to detect; there are no verification tools available for certain development tools/languages.

3.4.6.3 Ensure verification system is available and conduct verification

Verification should be done throughout the life cycle, following the verification strategy and plan. Ensure that the system, to perform verification, is available when needed (and potentially earlier for risk management):

- Generic verification tools that may be used in special ways for assurance or to help enhance assurance (e.g., generic bug-finding tools, testing that covers all branches, functional tests created to verify assurance assumptions).
- Security-oriented static analysis tools, e.g., those focused on identifying vulnerabilities. Many of these specifically search for common vulnerabilities (e.g., for software, those identified in the CWE). These may analyze binary and/or source.
- Security-oriented dynamic analysis tools. These include fuzz testers, which send large quantities of "random" data to see if the system fails in some catastrophic or insecure way. This may also include creating specialized attack scenarios as tests.
- Penetration testing and red team (aka security assessment). These involve teams to examine and attempt to penetrate the system. Identify procedures needed. Determine what is and is not in-bounds for such verification (e.g., is social engineering permitted), and how the issues not in bounds will be addressed instead. Minimize the limitations placed on red teams and penetration tests—since such limitations decrease the utility of the results—and ensure that those limitations are communicated only to those who need to know. After all, attackers do not limit themselves in the same way.
- Peer review and desk checking. Ensure that assurance-specific issues are covered in peer reviews, including that any checklist used includes such issues. These issues should cover common design weaknesses and language-specific dangerous constructs, such as handling malicious inputs, preventing or handling buffer overflows, etc.
- More specialized verification tools may be used, particularly for high assurance, such as theorem checkers/provers and formal model checkers.

For additional information on verification for software assurance, see the SwACBK(Redwine 2006). Verification should include examination for maliciously inserted functionality. Ensure that what is verified is the element to be actually used in the system (e.g., a malicious developer may attempt to ensure that what is verified is not what will be used). Verification should also examine that the constraints/assumptions on design are actually met if they are required for assurance.

3.4.6.4 Document and deliver verification data

Document and deliver the verification data as objective evidence supporting the assurance argument. The set of evidence, once collected and correlated, should support the broader assurance argument(s) and system assurance claims. This evidence should include the system

requirements verification matrix. This evidence should be considered part of the system, and as such, must have commensurate protection, e.g., from tampering, misuse, and/or hijacking.

3.4.6.5 Building the assurance case

While performing verification, update the assurance case to collect the evidence needed to justify the assurance claims:

- Demonstrate that the tools and techniques used to gather evidence are adequate to do so.
- Collect all static analysis and peer review results that support the assurance case.
- Collect test plans, approaches, and the associated test results that support the assurance case; these will typically be those for the critical elements (including defensive functions).
- Collect all penetration/red teams' test scenarios and results.

3.4.7 Transition

Transition (ISO/IEC 15288:2008, Section 6.4.7) is the process of establishing a capability or service into an operational environment. This process must ensure that the system once transitioned is assured, and does not have a negative impact on the assurance of other systems in the operational environment.

3.4.7.1 Prepare a transition strategy

The transition strategy must consider those aspects necessary for system assurance, as well as prevent other systems at the site from being breached. Issues to be considered include:

1. Configuration of the system in its target environment; consider the integration required with operational enterprise services such as an existing PKI.
2. Assessment and documentation of system assurance impact of any changes made to the system or environment.
3. Test and evaluation of the system as installed at the site with a focus on validation of the system in operation, considering other systems' impact on assurance. For example, consider the impact of the underlying operating environment or the enterprise services provided such as I&A.
4. Training of users, operations staff, and maintenance developers on the aspects of the system necessary for assurance. Consider the training implications for the system interoperating with any SoS, enterprise, or any overarching supporting enterprise services.

3.4.7.2 Prepare operational sites

The operational site must be prepared to receive the system. This may involve the adaptation of the operational site. Ensure that any changes made in support of this adaptation do not negatively impact assurance of the system being deployed, the other systems, or the infrastructure of the operational site. Examples include the installation of upgrade patches to the other systems at the site, or adding network connectivity to the site, either of which can introduce system vulnerabilities. A site survey could also be included prior to delivery to facilitate transition planning. Ensure that any downtime or parallel use during installation does not expose the site to additional vulnerabilities.

3.4.7.3 Deliver the system for installation

The delivery process for the system, and upgrade/patches, must protect the system from tampering, fraud, and exfiltration. For information this is often done through cryptography. The system elements must be protected through the intermediate stages of delivery. This includes such events as tampering of the system while in temporary storage or being moved between development and operational site.

3.4.7.4 Install, demonstrate, and activate the system

During the installation process, the system must be hardened to resist attack. This includes the disabling of unnecessary services as well as leveraging and enabling enterprise-secure mechanisms in support of the system and defensive functions. Where there are choices, use the more secure mechanisms.

When installing patches and upgrades, consider the implication of these changes to the assurance of the system. For example, the operation of the various security mechanisms and defensive functions in support of system operation may be disabled or degraded through patching.

Test and evaluation of the system must include the test and evaluation of its assurance. Validate that the system's assurance is not reduced as it is interfaced to the user/operational site. Demonstrate that the services provided by the system are sustainable by the enabling systems in the operational environment.

Train the users, operations staff, and maintenance personnel to execute and maintain the system's assurance. Keep an open communication channel between the development organization, transition, operational, and maintenance teams in support of assurance.

After activation, validate and demonstrate that the system assurance has been maintained. For example, the system and its surrounding subnet may be reexamined through the use of network, port, and vulnerability scanners to confirm compliance with security configuration policy in support of the system's assurance.

3.4.7.5 Record data

Ensure that transition information is analyzed for anomalies that may indicate attacks or vulnerabilities. Record and protect such data commensurate with its value to the system and its potential for abuse by attackers. For more information, see Constantine and Ambler (2002), Transition Approach Document, Office of Management and Budget, Financial Management Line of Business Taskforce Enterprise Architecture Working Group (EAWG 2004) and Pigoski and Sexton (1990).

3.4.7.6 Building the assurance case

While performing transition, environmental chances must be scrutinized to ascertain whether they affect the validity of the assurance case's evidence, arguments, and claims. Update the assurance case to complete the evidence needed to justify the assurance claims and arguments:

- Include evidence that any changes made (to adapt the operational site) have a positive or neutral effect on assurance (3.4.7.2) of the system or on other interconnected systems.

- Verify that the installed system is genuine and has not been tampered with (3.4.7.3). Such evidence could include chain of custody, intact tamper-evident packaging, and original manufacturer's mark (e.g., hologram).
- Verify that patches and upgrades, if any, have a positive or neutral effect on assurance (3.4.7.4) of the system or on other systems at the site.
- Verify that tests relevant to assurance achieved satisfactory results (3.4.7.4).
- Verify that staff training is adequate (3.4.7.4).
- Analyze anomalies in transition information (3.4.7.5) to determine that they have acceptably low risk.

3.4.8 Validation

The Validation process (ISO/IEC 15288:2008, Section 6.4.9) confirms that the system complies with stakeholder requirements when used in its intended environment, based on objective evidence. Validation is performed throughout the development and operation of the system to ensure that the 'right' system is being built, and is often accomplished through operational testing and evaluation. See also the Verification section, since much of the objective evidence generated by verification feeds into validation. Assurance needs must be validated, just as other needs must be validated, throughout the life cycle.

See also the Requirements and risk management section of this document. For additional general information about validation, see documents such as ISO/IEC 15408, NIST SP 800-53 and -53A, and DoDD 8500.01E.

3.4.8.1 Create validation strategy and prepare validation plan

The validation strategy and plan must be enhanced to address assurance issues. The validation strategy and plan would often be iteratively developed in parallel with the system requirements. Validation should occur throughout the life cycle, not just once after verification "completes," because problems are less expensive and faster to fix when discovered earlier. There may be additional stakeholders who are specifically concerned about system assurance; see the Stakeholder Requirements section. There may be a need to modify the scope of relevant teams (e.g., Integrated Product Teams) to address assurance issues, and those teams will require appropriate representation.

Necessary representation may be difficult to obtain in a timely manner, since assurance skill sets are a limited resource. The strategy and plan will need to address this constraint.

Determine who is responsible for validating the impact of system assurance from COTS elements and/or suppliers of concern, and how this validation will be performed. When the program involves SoS and/or FoS considerations, include in the assurance strategy distributed testing and other appropriate planning for validating the security of the architecture and SoS/FoS interfaces.

The strategy and plan must ensure that the validation process (e.g., by performing a temporary installation of the system) will not interfere with the existing environment, including the assurance of that environment. For example, if the system being validated receives an asset such as private keys, passwords, etc., these assets must be protected and/or destroyed

appropriately. (See also Section 3.1, Agreement Process, and Section 3.4.3, Architectural Design, for a discussion of alternatives during the acquisition process.)

3.4.8.2 Ensure that operators and facilities are ready for validation

Ensure that there is appropriate access to relevant information, including physical and logical access, for the validation process. This can especially be a problem where assurance information (such as threats and risks) is highly confidential.

Necessary representation (including operators) may be difficult to obtain in a timely manner because skilled personnel may be a limited resource. Any system has assumptions about its environment, and these assumptions should be validated and revalidated (as the environment changes) so assurance is maintained. Ensure that the enterprise environment is prepared for such validation. Special arrangements may need to be made for testing assurance assumptions, if they occur in the production environment, as this testing may otherwise appear to be an attack on the enterprise.

3.4.8.3 Conduct validation

Conduct validation, according to the validation strategy and plan. When conducting validation, include assurance considerations. For example, review user needs as follows:

- Who are the potential attackers, and how important is subversion of this system to them? Who would the users not trust to control this system?
- What would the attackers' goals be?
- What are the services and their importance/prioritization/criticality to the user? What is the significance of service failure if it is caused by an attack?
- What new attacks and/or previously unanticipated attacks have been identified (including information on similar systems, if any)?
- What impact will new usage modes and models have on system assurance?
- What anomalous behavior has been observed, and does it suggest an impact on system assurance?
- How long can users live without that service, or specific parts of the service?
- How much degradation in service, and which service, can the user live with?
- What is the impact of the systems supporting infrastructure on use and logistics?
- Which reused elements (including COTS) or suppliers raise concerns?

Validate that the system is "secure by default" and "secure in deployment" in its expected environment(s). As changes occur (e.g., in usage modes and environment), ensure that the system and its assurance is revalidated.

The system life cycle and operational environment are dynamic, many systems are complex, and many attackers are intelligent and adaptive. This may result in many "unknown unknowns" that impact system assurance. Thus, "unknown unknowns" must be identified and addressed throughout the system life cycle. Keep track of any new attacks coming in on honeynets, etc. Ensure that networking is being done with System Administrators to see if any anomalous traffic is noticeable that might affect the system. When a previous "unknown unknown" is identified, its

assurance impacts must be reassessed and validated, consistent with the resources available and risks to the system.

In some cases modeling and simulation may be useful for validating assurance. This may be accomplished via techniques such as system mock-ups/screenshots and animation of formal models.

3.4.8.4 Isolate the part of the system giving rise to non-conformance

Isolate not only the specific requirement/design/implementation issue that is the root cause of the non-conformance (and thus needs correction), but also identify why and how that non-conformance occurred to facilitate corrective action to prevent its recurrence. Do this to (1) detect similar problems so they can also be detected and corrected, and/or (2) develop preventive measures where possible to prevent its reoccurrence (e.g., through process improvement, or a change in materials, tools, or technology).

When an assurance-based non-conformance is identified, a revalidation process should be used to determine the priority of the service, based on user needs, and determine how to feed that information into the life cycle (e.g., through element change, isolation, and so on). This should include a determination of the impact on the system assurance case with regard to the non-conformance.

Isolating a part of the system for an assurance non-conformance may be difficult, because assurance is an emergent property. An emergent property is a property of the entire system working together, not solely of one particular element. Often assurance issues do not have a single cause, but instead are caused by a combination of factors and system elements, and may involve a chain of many elements. The visible effect may be far removed from the initial causes. The assurance case may help guide identifying the factors involved and what to repair. In some cases it may be necessary to change multiple elements to establish or re-establish assurance.

3.4.8.5 Analyze, record, and report validation data and make it available

Document and deliver the validation data as objective evidence supporting the assurance argument. The set of evidence, once collected and correlated, should support the broader assurance argument(s) and system assurance claims. This evidence should be considered part of the system, and as such, must have commensurate protection, e.g., from tampering, misuse, and/or hijacking. Some assurance information may need to be confidential and specially protected, since it provides information on the system's weaknesses to attackers.

3.4.8.6 Building the assurance case

During validation, an argument supporting the claim that the system possesses the properties required to meet user security requirements should be developed. Evidence to support this argument should be collected and/or referenced, as well. Update the assurance case to complete the evidence needed, including:

- Data that support the argument that the system meets the stakeholders' security requirements. (3.4.8).
- Evidence that revalidation has been performed successfully after encountering non-conformance, or when there have been changes in the environment, or in the use of the system (3.4.8.3, 3.4.8.4).

3.4.9 Operation (and Training)

The point at which a system transitions to operation is critical for system assurance success. Often, the engineering team members with closest knowledge of the system and its inner workings are migrating away from the program (ISO/IEC 15288:2008 section 6.4.10). New team members, many of whom will be held accountable for the correct functioning of the system during operation, are just entering training programs and learning how to operate and use the system. Thus while ISO/IEC15288:2008 section 6.4.10 is titled Operation, we are including "and Training" to emphasize the importance of training. This emphasis will normally require two forms of adjustment: (1) adjusting the mission process to successfully use the system and (2) modifying the system itself, perhaps despite its implementation, to make it usable by the mission. Modifications may encompass the addition of services, updates, upgrades, changes in configuration, interfaces, etc. The implications of new team members, for example, will have to be recorded and the impact on system assurance considered. All the work to this point will either be fulfilled in assured operation, or compromised through incorrect usage.

3.4.9.1 Training users

The implementation of risk mitigations and assurance tools does not eliminate the need for training. Testers may not understand the results of a tool or how to apply those results without first understanding the kinds of vulnerabilities the tool is addressing. A training plan should be included as part of the strategy in support of systems life cycle support.

Operating a system in an assured manner requires users who have been properly vetted and trained. The system risk assessment should have identified the potential impact of risks from inadequately trained or untrustworthy users.

User training must include information on how to establish and maintain system assurance. The scope of training shall encompass information about the system and its relationship to the operational environment, so that users can maintain system assurance and leverage existing services to do so (e.g., security services, firewalls, and so on). The training shall include both positive aspects (what to do) and negative aspects (what not to do) and why, including avoiding or countering social engineering attacks. Training should include information on what the system is expected to do in typical usage circumstances, in order for users to identify when the system does something unexpected (because this may indicate a potential security breach). Training should include how to handle important non-routine and emergent conditions so users do not compromise system assurance. Training should include how users should report or act upon such non-routine or emergent conditions.

To operate the system, these vetted and trained users must be granted the privileges necessary to perform their roles. This is typically done through some sort of I&A system, then tying those identities to authorizations, per the associated policies (system, security, and enterprise policies). For system assurance to be achieved, some roles may need to be separated, and/or in some cases two-person control may be required.

3.4.9.2 Monitor the system

While monitoring the system, the objective is to ensure that the system is operating securely, as intended. The requirements for monitoring will be determined by the system risk assessment,

and may include monitoring for intrusion and monitoring for deviations from expected operation or configuration that affects system assurance. For example, if a system is not supposed to be connected to the general-purpose Internet, passive monitoring for general-purpose network addresses and/or periodic attempts to access the Internet can be used to ensure that this is always the case. Monitoring can employ different tools and techniques which are available (e.g., intrusion detection tools, network mapping tools (to detect unexpected changes), risk assessment, and audit log correlation tools).

3.4.9.3 Modifying the system

For the duration of system operations, the system may need to be modified (e.g., because of changes in need or because of system problems, such as vulnerabilities). Such modifications may be deployed through system upgrades, updates, or changes in fielded configurations. Regardless, a risk assessment or security impact analysis of the changes should be performed in order to ensure system assurance can be maintained.

Because timely modifications to the system may sometimes be impractical, this can mean continued and perhaps unacceptable exposure. For example, a shipboard system might not be modified until the ship returns to port, or it might be performing other critical operations that cannot be interrupted. Deriving several alternatives for handling such an exposure may be required, including working around a problem, or informing the user how the remaining exposure may impact their mission. For all these alternatives, perform an analysis of the impact on system assurance.

System assurance is not static, because usually neither the system nor its mission is static. Information on system modifications and how they impact the system's assurance, must cycle back into the appropriate engineering team (e.g., development, maintenance, etc.). The arguments and evidence that justify system assurance deteriorate over time unless maintained during operational adjustments.

3.4.9.4 Building the assurance case

Update the assurance case to complete the evidence needed to justify the assurance claims and arguments:

- For all aspects of the assurance case that require training of the operations and sustainment staff, collect, maintain, and audit training records and other documentation to justify that appropriate training is occurring periodically.
- Perform regular audits of operation records, to verify that there is no evidence that the system's assurance has been unknowingly subverted.
- Periodically verify that environment assumptions necessary for the assurance case are still valid.

3.4.10 Maintenance

The maintenance process (ISO/IEC 15288:2008, Section 6.4.11) sustains the capability of a system to provide its service. It monitors system operations and records problems, acts on them to take corrective/adaptive/preventive actions, and confirms restored capability. To this end, the process involves logistics, system interfaces to field support equipment and other systems, system

fault detection and isolation, facilities assessment, computer resources life cycle support, and so on. The maintenance process should sustain, and in some cases improve, system assurance.

3.4.10.1 Maintenance strategy

Prepare and document a maintenance strategy, including schedules and resources required in order to perform corrective and preventive maintenance. This strategy can be part of the updated project plan or a separate document to be included as part of the maintenance process input package. Responsibilities for assurance should be purposefully and appropriately assigned; typically this will be across multiple individuals and multiple disciplines.

The maintenance or system support strategy should be generated early during planning, so that maintenance constraints can be provided as inputs to the requirements process, and to accommodate sustainment issues associated with the intended operational environment and/or interfacing systems. It should provide part of the framework upon which systems engineering and logistics planning are developed. This strategy should be refined throughout the life cycle as more detailed information becomes available.

The maintenance strategy defines the schedules and resources required to perform corrective and preventive maintenance in conformance with availability requirements. It should include:

1. The corrective and preventive maintenance strategy to sustain service in the operational environment in order to achieve customer satisfaction. This should update documentation as necessary. Corrective maintenance may be identified through modeling and simulation performed during a repair level analysis. (See also Section 3.4, Technical Processes.)
2. The scheduled preventive maintenance actions that reduce the likelihood of system failure without undue loss of services (e.g., suspension, restriction, or reduction of the services provided (to ease assurance)).
3. The number and type of replacement system elements to be stored, their storage locations and conditions, their anticipated replacement rate, their storage life, and renewal frequency. For assurance, identify how they will be protected while they are stored, and how retrieval of them is managed, controlled, and logged.
4. The skill and personnel levels required to effect repairs and replacements. For assurance, determine how to ascertain their trustworthiness, how to ensure that only authorized individuals can perform repairs/replacements, and include a process to audit deviations. Explain how they will receive sufficient training.
5. Configuration management maintenance approach; see Section 3.3.6, Configuration Management.

The strategy should define processes for developing, verifying, and delivering changes that are protected from unauthorized and undetected changes, and ensure that recipients can verify that changes they receive are authorized.

3.4.10.2 Define the constraints on system requirements

Per ISO/IEC 15288:2008, constraints on system requirements are unavoidable consequences of the maintenance strategy.

In some cases it may be necessary for a fielded system to authenticate an update system or vice versa. If the system can be updated in the field, it may be necessary to include a public key in

the deployed system that can be used to cryptographically verify that updates are from a trusted source. In addition, it may be necessary to have a process for providing a private key to deployed systems, so that their identity can be verified. There may be a need for some sort of "call home" update system, or conversely, it may be necessary to disable such a system. Evaluate this function and its impact and need for the service on the system. Physical security measures (e.g., holographic images embedded in hardware elements) may be necessary to prevent use of counterfeit parts.

If maintenance must occur with little to no service interruption, identify what maintenance must occur under these conditions, and how much service interruption is acceptable. Often this is done with alternative/backup systems, processes, or pathways; these must be examined to ensure that their implementation and use do not introduce vulnerabilities.

There may be requirements that the system support special maintenance roles, which may be limited in their actions. It may be necessary that the system not be able to perform certain actions because it is physically exposed to untrustworthy parties. An "undo update" function may be important to restore functionality, should an update cause critical functionality to break.

3.4.10.3 Obtain the enabling systems, system elements, and services used during maintenance

As part of assurance maintenance, consider including products and services that monitor/maintain assurance of the elements (e.g., intrusion detection, configuration management audits, audit log examination, risk assessment).

Ensure that any infrastructure used to distribute software or data updates can withstand attack, and is able to scale to the number of users.

If specialized equipment or tools are required for retirement of elements during maintenance (e.g., incinerators for storage devices), obtain them.

3.4.10.4 Implement problem reporting and incident recording

There may be a need for a special problem-reporting procedure for vulnerabilities, to prevent such information from reaching potential attackers (e.g., through cryptography), and to ensure that critical vulnerabilities can be considered immediately.

Some incident recording may need to be specially protected from arbitrary access, since it may reveal significant vulnerabilities or weaknesses in a system.

In some instances, a community (e.g., banking, government, etc.) may require notification of certain types of vulnerabilities or weaknesses (depending on their impact), expected corrective actions, and associated timelines, as a cost of doing business.

Problem reports must be prioritized based on impact; those with a significant impact on assurance should have high priority. Techniques such as using vulnerability trees may aid in this analysis.

3.4.10.5 Implement failure identification

Correction procedures should consider if the reported problem is evidence of an unknown vulnerability and/or of an attack on the system. It may also be evidence of non-compliance with a security policy or controls (e.g., configuration management controls).

Identify the failure and its associated fault. Determine the fault's impact (i.e., through impact analysis and/or risk assessment), including a determination of whether it is an exploitable vulnerability. In some cases it may be difficult to determine if it is exploitable; if this is uncertain, it should be treated as an exploitable vulnerability.

When a fault is detected, an assessment of the fault should be performed to identify whether it poses a threat and to diagnose its origin. Where appropriate, forensic analysis of the failure should be performed to identify the root cause and to take appropriate action.

If a new failure is identified, correction procedures should involve looking for similar or linked constructs to also repair. This is because a type of fault found in one place may have occurred in other places as well and may have also previously escaped detection. Root cause analysis of such new vulnerabilities is one approach for performing this search. It is also imperative that when exploitation is identified, prompt action is taken to mitigate the consequences among all affected areas.

3.4.10.6 Implement correction procedures

Develop a plan to correct the product, and if necessary, communicate it with relevant parties. In some cases, some functionality may need to be removed to maintain assurance; this situation in particular may require such communication. In some cases, there may be conflicting goals (e.g., functionality vs. assurance, cost vs. speed of change); see Section 3.3.5, Risk Management, for how to adjudicate such issues.

Verify, using test and evaluation, for example, that the corrective action is effective. The correction procedures should not introduce new vulnerabilities or weaknesses, and should avoid or minimize either reducing or interrupting service. The corrective actions should also update any relevant documentation, including the justification and supporting evidence of assurance.

3.4.10.7 Confirm logistics actions satisfy replenishment levels

Based on the maintenance strategy, ensure that spare elements will be protected while they are stored, and that retrieval of them is managed, controlled, and logged. This could include fail-over systems residing on a network ready for operation, as well as substitute elements not yet installed.

3.4.10.8 Perform preventive maintenance

Ensure that software systems are kept up-to-date on their security patches, where practical, instead of waiting for attackers to subvert the system.

3.4.10.9 Maintain a history

This historical information may be sensitive, and as such appropriate measures may be necessary to protect the confidentiality, integrity, accountability (including non-repudiation), availability, and auditability of the data. For more information, see the following:

- ISO/IEC15288:2008: Systems and Software engineering–System Life Cycle Processes
- IEEE P1633\AIAA R-013A Standard for Software Reliability
- NASA-STD-3000 (NASA 1995)
- Andrews, Richard A. Maintenance Planning (paper). Air Force Institute of Technology.
- Department of Defense Handbook for Acquisition Logistics
- Office of the Assistant Secretary of Navy (RD&A) Acquisition Logistics for the Rest of Us, an Assessment Tool from A to IOC
- Defense Acquisition Guidebook

3.4.10.10 Building the assurance case

Changes to the system during system maintenance may have wide-ranging (little to significant) impact on the assurance case. Since each change can involve one or more of the technical processes described in the preceding sections, the maintenance of the assurance case involves following the guidelines for the processes involved in the change. For example, a change which involves a change to code with no change in requirements or architecture design requires the use of the guidelines for the implementation, integration, verification, transition, and validation processes (3.4.4 thru 3.4.8).

During maintenance, update the assurance case to complete the evidence needed to justify the assurance claims and arguments:

- For each change, perform the activities in the applicable processes as described in this guidebook (see sections 3.4.1 - 3.4.9), including updating the assurance case as appropriate.
- Analyze the defect and enhancement requests to determine any deviation that affects the assurance case, or trends that may impact the assurance case (e.g., through threats, vulnerabilities, and security breaches).

3.4.11 Disposal

The disposal process (ISO/IEC 15288:2008, Section 6.4.12) deactivates, disassembles, and removes the system and/or system elements and/or any waste products. The process destroys, stores, or reclaims system elements (including software, hardware, and data). For assurance, procedures must be defined for the safe and secure sanitization and destruction of the system elements, along with ensuring data are destroyed or migrated safely and securely. For example, this may include the destruction of private keys embedded in cryptographic modules.

3.4.11.1 Disposal strategy

In addition to environmental concerns, some software, hardware, and/or data may need to be destroyed or retained. There may be restrictions in terms of who may handle data (e.g., handling caveats for classified information or personally identifying information), and/or there may be a need to destroy hardware, software, or data to prevent reverse engineering, reuse, or potential release. Some elements may need to be handled specially (e.g., cryptographic gear may need to be destroyed to prevent reverse engineering, or have embedded keys that must be erased using special processes). Additionally, some elements may be released for reuse or resale; in these cases, elements must be examined to ensure that software, hardware, or data that should not be

released are not thereby accessible or recoverable. For example, if a computer system or storage device is resold, ensure that the storage device(s) is properly erased before release. Laws, regulations, policies, licenses, contracts, and other requirements may impose specific disposal criteria (requirements or restrictions) related to assurance.

A disposal strategy should be developed that identifies the elements (hardware, software, and/or data) to be disposed of, and the approved process for disposal. Particular focus should be placed on the disposal of more sensitive elements or information, system CPIs including classified information, elements that must not be reverse-engineered, and personally identifying information.

One strategy may be to keep a careful record of system elements to ensure that proper techniques are applied during disposal; see Section 3.3.6, Configuration Management.

The disposal strategy should:

1. Sanitize/destroy hardware, software, and data, as appropriate, in all locations it resides. This may be determined by the licensing or other legal agreements. Should some elements (e.g., storage devices and computing equipment) be resold or reused, the licensing may forbid transfer of software or certain software upgrades. This is particularly the case with enterprise-wide software license agreements. Disposal of an operating system (OS) is often tied to the computer hardware on which it was originally installed. When an enterprise donates a computer to a charity or sells it to a third party, often the OS also must be donated.
2. Sanitize/destroy all associated media as appropriate; this includes original media used to load the software or data, backup copies of software or data, and portions of the software or data that may exist in system temp or swap spaces.
3. Sanitize/destroy associated technical manuals and training materials for the hardware, software, and data, as appropriate.

Requirements for clarifying destruction and licensing policies need to be established. Policies and procedures provide standard methodologies for destruction and control the risks associated with improper disposal of software and computing equipment.

The strategy may need to be reviewed by the organization’s legal counsel, and include a process to track and record compliance. It may be necessary to keep an inventory of serial numbers, dates of purchase, dates of destruction, and means of destruction. By providing a complete disposal audit trail, it is possible to assure that hardware, software, and/or data were not inadvertently exposed.

3.4.11.2 Destroy the system

There are many different elements that may need to be destroyed, or have software or data removed from them. For example:

- Any hardware element that should not be reverse-engineered.
- Magnetic hard drives.
- Flash memory. Note that “overwriting” to flash memory typically does not actually overwrite all the underlying information, due to wear leveling and garbage collection mechanisms built into these devices.

- Volatile memory. Volatile memory elements do not normally retain data after removal of all electrical power sources, and when reinserted into a similarly configured system, do not contain residual data. However, volatile memory may become non-volatile through unknown battery backups, and a highly resourced adversary may be able to retrieve some information.
- Nonvolatile memory elements do retain data when all power sources are discontinued. Nonvolatile memory elements include Read Only Memory (ROM), Programmable ROM (PROM), or Erasable PROM (EPROM and EEPROM). Examples of their use include BIOSes, FPGA configurations, and ASICs.

All of these elements may reside on boards, modules, and sub-assemblies. Naïve approaches to removing software and data often fail. Simply deleting files from the media, or reformatting the media, is often insufficient to completely erase data so that it cannot be recovered; often such data is easily recovered. Approaches for destroying software or data include:

1. Overwriting the data. Overwriting of data means replacing stored data on a drive with other data. There are numerous software products that rewrite media to wipe off deleted files or the entire contents of a computer drive. Many of the products have various levels of overwrite, so be sure to select the option that is compliant with your requirements. As noted above, this often fails with flash memory.
2. Degaussing. This is a procedure that reduces the magnetic field on media virtually to zero by applying a reverse-magnetizing field. Properly applied, degaussing renders any previously stored data on magnetic media difficult to read. Magnetic media can be divided into three types (I, II, and III) based on their coercivity, which defines the magnetic field necessary to reduce a magnetically saturated material's magnetization to zero. The level of magnetic media coercivity must be ascertained before executing any degaussing procedure.
3. Physical force (e.g., pounding with a sledgehammer). Bending, disfiguring, or otherwise mutilating the media may render its contents unrecoverable, but it may be ineffective for disposing data. It is not typically regarded as a best practice.
4. Incineration. This is typically the best way to destroy data (including software) or hardware.

Highly resourced adversaries may be able to recover data from overwritten, degaussed, or physically damaged devices, so processes such as incineration may be necessary in such cases. In some cases, an independent party may need to witness the destruction.

3.4.11.3 Outcome/benefits

Outcome of performing proper destruction of systems is elimination of the risk associated with access to residual hardware, software, or data on the system or its elements. This reduces the risk of revelation of system information that might enable an attacker to penetrate the system, illegal distribution of licensed software, release of sensitive system information to an unauthorized organization or individuals, enabling reverse engineering, etc.

For more information on disposal, see the following:

- DoD Directive 5220.22 (2004, certified 2006), "National Industrial Security Program"
- DoD 5220.22-M (2006), "National Industrial Security Program Operating Manual"
- NIST 800-88 (2006), "Guidelines for Media Sanitization"
- ISO/IEC 9126, Parts 1, 2 & 3. Software Engineering – Product quality

- ISO/IEC 12207 Software Engineering – Software life cycle processes
- ISO/IEC 14598, Parts 1-6. Information Technology – Software product evaluation.
- Software Disposal: Old Software Never Dies (O'Brien and Park 2003)
- Equipment Disposal Policy (Draft) (Gannon University 2007)
- State of Kentucky Auditors Alert on the Sanitation of Electronic Media (Luallen 2005)

3.4.11.4 Building the assurance case

Update the assurance case to complete the evidence needed to justify the assurance claims and arguments:

- Verify that all elements requiring disposal are appropriately identified and destroyed in an appropriate manner. Note: For highly critical elements, this often this requires incineration.

4 SYSTEM ASSURANCE GUIDANCE IN U.S. DEPARTMENT OF DEFENSE PROGRAMS

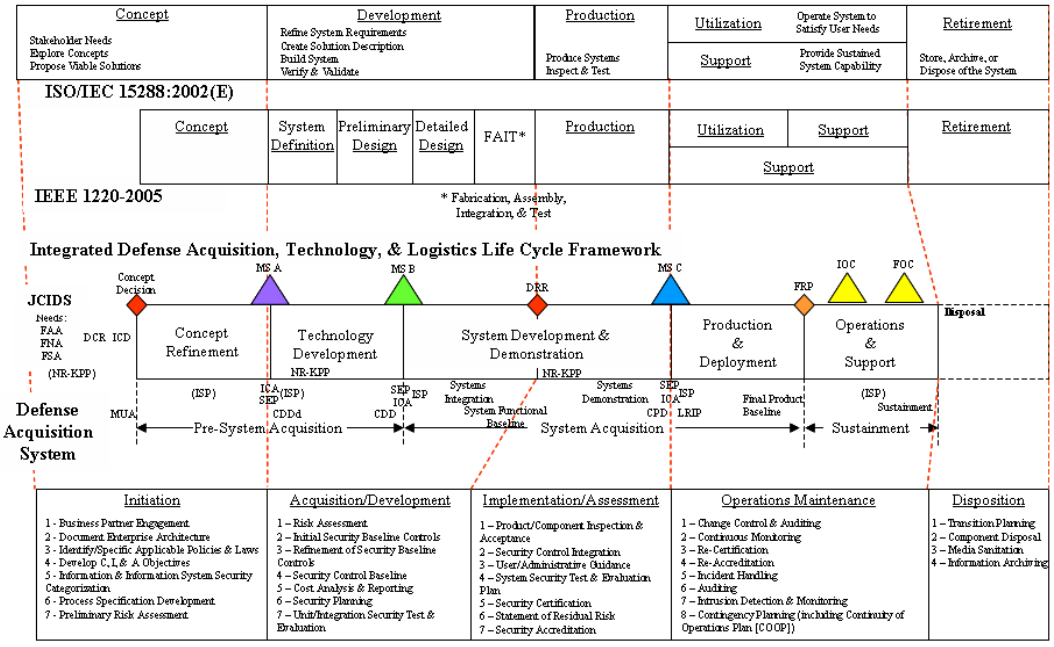
This section contains system assurance guidance specific to U.S. Department of Defense (DoD) programs, for use by the DoD, its contractors and subcontractors. Future editions of this guidebook may contain system assurance guidance for other specific industries or areas.

4.1 Introduction

This section is organized according to the Integrated Defense Acquisition, Technology and Logistics Life Cycle Management Framework. This section also follows DoD Directive (DoDD) 5000.l, DoD Instruction (DoDI) 5000.2, and the DAG, and uses these documents' terminology. The goal of this section is to enable the DoD to acquire or produce assured systems. These systems are inclusive of both weapon systems and IT systems. This section discusses the topics that should be addressed during the phases of the DoD integrated defense acquisition, technology, and logistics life cycle management framework. As discussed in DoD Instruction 5000.2 and the DAG, the life cycle phases include:

- Concept Refinement
- Technology Development
- System Development and Demonstration
- Production and Deployment
- Operations and Support

Figure 4-1 shows how this framework relates to industry systems engineering and information security standards.



Source: DAG

Figure 4-1 DoD Life Cycle Framework and National Institute of Standards and Technology Information Security and the System Development Life Cycle

The bulk of this section is subdivided into DoD review milestones and milestone decision points from this framework. For each review (including the system engineering technical reviews (SETRs)), the DAG description is quoted, followed by a list of the most important system assurance items to complete prior to the milestone. For each review a specific cross-reference to Section 3 for technical instruction is also provided. Where there are assurance activities that cannot be associated with specific reviews, additional subsections are added to contain them. Note that this section not only focuses on the DoD, it makes the assumption that the audience includes system integrators providing systems (both IT and warfighting) to the DoD environment. The following table shows how the technical processes in this guidebook Section 3.4 (based on ISO/IEC 15288:2008) map to the DoD process in the DAG:

Table 4-1 Map of Guidebook Technical Processes, ISO/IEC 15288, and Defense Acquisition Guidebook

Guidebook (Section 3.4)	ISO/IEC 15288 (Section 6.4)	Defense Acquisition Guide (Chapter 4)
3.4.1 Stakeholder Requirements Definition	6.4.2	4.2.4.1 Requirements Development
3.4.2 Requirements Analysis	6.4.3	4.2.4.2 Logical Analysis
3.4.3 Architectural Design	6.4.4	4.2.4.3 Design Solution
3.4.4 Implementation	6.4.5	4.2.4.4 Implementation
3.4.5 Integration	6.4.6	4.2.4.5 Integration
3.4.6 Verification	6.4.7	4.2.4.6 Verification
3.4.7 Transition	6.4.8	4.2.4.8 Transition
3.4.8 Validation	6.4.9	4.2.4.7 Validation

This section begins by addressing assurance as it relates to CPI and the Program Protection Plan (PPP) for achieving protection of critical technologies, and confidentiality of associated information. This guidebook expands their use beyond such critical technologies, to include protection of critical functionality. Although CPI and PPP are mandated only for critical systems, the guidebook recommends the use of the PPP process for all DoD systems. This is followed by anti-tamper and supplier assurance subsections.

Note that it is DoD policy to manage all interconnections of DoD information systems “to continuously minimize community risk by ensuring that the assurance of one system is not undermined by vulnerabilities of interconnected systems” (DoDD 8500.1 section 4.14). Community risk is the “probability that a particular vulnerability will be exploited within an interacting population and adversely impact some members of that population.” (DoDD 8500.1 section E2.1.5). System assurance activities reduce community risk.

To implement system assurance, existing DoD processes and deliverables can be used for defining and recording the information required by the following subsections. The goal is to make existing processes and deliverables more effective by addressing the system assurance threats. For example, results of static analysis may be reported in review results and/or test results deliverables.

The assurance guidelines below help programs practice various approaches for delivering assurance, such as employing the tenets of defense-in-depth (DoDD 8500.1, Section 4.4 and 4.7; DoDI 8500.2 E3.2.4.2).

DoDD 8500 is a controls regime driven by mission assurance category (MAC) and confidentiality levels (CLs); MAC and CL unambiguously determine IA requirements. DoD requires all DoD systems to directly or indirectly (e.g., through inheritance from other systems) meet all information assurance controls within DODI 8500.2 identified by their MAC and CL in order to meet accreditation requirements.

Where this document calls out particular IA controls or sets of controls for attention, it does suggest that all IA controls need not be complied with, but focuses on the IA controls that affect system assurance or how information assurance controls can be effectively addressed within the Defense Acquisition Lifecycle. DoDI 8500.2 controls listed in enclosure 4, attachments 1 and 4 (A1 and A4), were used; as noted in DoDI 8500.2 table E4.T2, these compose the most stringent and complete set of controls.

Relevant DoD directives and instructions include the following:

- DoDI 8500.2, Information Assurance (IA) Implementation, which implements DoDD 8500.01E Information Assurance (IA)
- DoD Certification & Accreditation (C&A) processes, including the DIACAP processes. (DoDI 5200.40, DoD 8510.1-M, and DoDI 8510.01)
- DoDD 5200.39, Security, Intelligence, and Counterintelligence Support to Acquisition Program Protection
- DoD 5200.1-M, Acquisition Systems Protection Program, which discusses Program Protection Plans (PPPs)

Key Hardware Assurance DoD Policy References include:

- Deputy Secretary of Defense Memorandum, “Defense Trusted Integrated Circuit Strategy” October 10, 2003
- Under Secretary of Defense (AT&L)/Assistant Secretary of Defense (NII) Memorandum, “Interim Guidance on Trusted Suppliers for Application Specific Integrated Circuits (ASICs),” January 27, 2004
- Under Secretary of Defense (AT&L) Memorandum, “Encouraging Industry Participation in the Trusted Foundry Pilot Program,” January 27, 2004
- Under Secretary of Defense (AT&L) Memorandum, “Expansion of Trusted Foundry Program,” January 27, 2004

4.2 Program Protection Implementation

Program protection is the “safeguarding of defense systems and technical data anywhere in the acquisition process to include the technologies being developed, support systems (e.g., test and simulation equipment), and research data with military applications. This protection activity involves integrating all security disciplines, counterintelligence, and other defensive methods to protect the essential program information, technologies, and systems data from intelligence collection and unauthorized disclosure.” (DoD Manual 5200.1-M).

Critical program information (CPI) is “critical program information, technologies, or systems that, if compromised, would degrade combat effectiveness, shorten the expected combat-effective life of the system, or significantly alter program direction. This includes classified military information or unclassified controlled information about such programs, technologies, or systems” (DoDD 5200.39). All programs require a CPI assessment, and when CPI is identified,

then a program protection plan (PPP) must be developed. Even if a program does not have a formal PPP, existing guidance for developing PPPs can be valuable for identifying potential threats and countermeasures. There is a pending policy DODI 5200.39, to replace DoDD 5200.39, that will clarify that the definition of CPI includes information, functionality, and technology. Historically, programs have often interpreted CPI as being limited to confidentiality. With this pending policy clarification, it is clearer that programs need to cover the confidentiality, integrity, availability, authentication, accountability (including non-repudiation), and auditability of critical program information.

DoDD 5200.39 explains that program protection planning is a "comprehensive effort that encompasses all security, technology transfer, intelligence, and counter-intelligence processes through the integration of embedded system security processes, security manpower, equipment, and facilities." In particular, it notes that the program protection plan should be a "risk based, comprehensive, living plan to safeguard critical program information (CPI)… [it] is used to develop tailored protection guidance for dissemination and implementation throughout the program for which it is created." This plan should cover the entire system life cycle.

PPP countermeasures are technical, procedural, or administrative means to deny access to CPI from identified threats and prevent potential compromise. System assurance mechanisms may be used as PPP countermeasures to protect CPI, as part of the system engineering life cycle.

To develop a PPP:

1. Identify Critical Program Information (CPI)

2. Request Multidiscipline CI (MDCI) Threat Assessment to identify possible threats

3. Identify vulnerabilities, and perform a risk analysis

4. Determine which PPP countermeasures to apply to the program (including anti-tamper and supplier assurance)

5. Set criteria and define processes to evaluate PPP countermeasure effectiveness

6. Summarize the results of this process within the PPP

When developing this plan, the following general process from DoD 5200.1-M on developing program protection plans may be especially helpful in achieving system assurance as PPP countermeasure:

- A program protection plan should safeguard information through the entire system life cycle (DoD 5200.1-M, C3.1). Traditional PPPs typically focus on confidentiality; they should also address issues of integrity, availability, authentication, and accountability.
- To identify what needs protecting, identify and set priorities on those operational or design characteristics of the system that make it unique and provide superior mission capabilities (DoD 5200.1-M, C3.1.1). From this, derive what information needs to be protected, and in what way, to preserve these capabilities (5200.1-M, C3.1.2).
- Determine where this information is stored and could be changed (5200.1-M C3.1.3).
- Identify the program's vulnerabilities to specific threats across the full life cycle, including intelligence threats (DoD 5200.1-M, C3.1.4 and C3.1.5). For assurance, consider insider threats, subversion via suppliers, and unintentional and intentional weaknesses being inserted into the system during its development.

- Identify the countermeasures to be employed (DoD 5200.1-M C3.1.6). Note that adversaries are active, not passive.
- Identify the costs (DoD 5200.1-M, C3.1.7).
- Identify elements that require classification (DoD 5200.1-M C3.1.8). See 5200.1-M Chapter 4 for more information on this.
- Identify risks and benefits of developing, producing, or selling the system abroad or to less-trusted organizations, as well as how to protect information in such cases where necessary (DoD 5200.1-M, C3.1.9). See 5200.1-M Chapter 5 for more information on this, as well as this guidebook's section on supply chain (below).
- Identify the design features or support equipment required to reduce operational security vulnerabilities upon deployment (DoD 5200.1-M C3.1.10). Review the details of this guidebook, particularly this section and section 3.4; see also 5200.1-M Chapter 6.
- Develop the resulting plan to protect the program, and then follow the plan in support of system assurance. Modify the plan as necessary throughout the system life cycle. See 5200.1-M Chapter 3 for approaches that may be helpful.
- Develop such a plan by coordinating with others who have the expertise necessary to implement the above from an assurance perspective, including those from intelligence, systems engineers, information security, safety experts, reliability experts, and others. See DoD 5200.1-M section C3.2.

The following table shows the DoD 5200.1-M general actions for program planning, the specific subsections of DoD 5200.1-M that provide more detail on how to perform them, and relevant sections in this guidebook that provide specific guidance on system assurance.

Table 4-2 Program Planning Actions in 5200.1-M and Guidebook

5200.1-M General Actions	5200.1-M Specifics	System Assurance Guidebook Reference
C3.1.1 – Id Priorities	C3.4	3.1.1, 3.3.7, 2.3, 3.4.1, 3.4.2.3
C3.1.2 – Id critical elements and information	C3.6	3.3.7, 3.4.2.3
C3.1.3 – Id Locations	C3.5, C3.7.2.4	3.1.1, 3.4.7, 3.4.9, 3.4.10
C3.1.4 –Id Threat	C3.8	3.3.5, 3.4.1
C3.1.5 – Id Vulnerabilities	C3.7	3.3.5, 3.4.1
C3.1.6 – Id Countermeasures	C3.9	3.3.5, 3.3.6, 3.4.3, 3.4.4, 3.4.5
C3.1.7 – Costs	C3.10	3.4.7 and references noted in C3.1.6
C3.1.8 – Classification	C4	References noted in C3.1.6
C3.1.9 – Export	C5	3.1.1, 3.1.2
C3.1.10 – Design	C6	3.4.3 – 3.4.11
Program Protection Plan itself	C3.3	3.1.1, 3.3.1, 3.3.4, 3.3.5
Coordination	C3.2	3.1.1, 3.3.1
Implement protection plan and manage it		3.3.2, 3.3.4

No particular outline for a plan is required; however, following is a sample plan outline, based on DoD 5200.1-M Acquisition Systems Program Protection, C3.3.5, that may be useful. Plans typically include some information by reference:

1. System Description. These are the system's high-level requirements. This would typically be a summary in a few sentences and a reference to a more detailed

system description. From an assurance perspective, the issue is what is unique to the system and needs to be protected in terms of confidentiality, integrity, availability, authentication, accountability (including non-repudiation), and auditability, etc. See DoD 5200.1-M, C3.4.

2. Program Information. Discuss the organization and structure of the program's acquisition chain of command and supply chain, identifying key contractors and suppliers (and how to contact them). See DoD 5200.1-M, C3.5.
3. Essential Program Information, Technology, and/or Systems (EPITS – today termed CPI). Identify the critical elements that must be protected, including their confidentiality, integrity, availability, authentication, accountability (including non-repudiation), and auditability. This can be done by using a functional decomposition (a standard systems engineering process). See DoD 5200.1-M, C3.6, and guidebook 3.3.7, 3.4.1, and 3.4.2.
4. Vulnerabilities. The scope of traditional program protection plans focus on confidentiality, but vulnerabilities should really be considered for the full gamut of system assurance. Vulnerabilities must be considered across the system life cycle, including operational, and across all aspects, including personnel and facilities. See DoD 5200.1-M, C3.7, and guidebook section 3.3.5 and 3.4.1.
5. Threats. This guidebook considers a wider set of threats than does 5200.1-M; the full range of threats to a system should be considered, so that they can be countered. These include threats from foreign governments gaining access to new technology or intelligence data, as well as subverting elements so that they will fail or be controlled by unauthorized parties. Potential threat agents include governments, terrorist organizations, organized crime, malevolent corporations, and individuals, all of whom can be considered active adversaries. See DoD 5200.1-M, C3.8, and guidebook 3.3.5 and 3.4. (Note: there is no current policy change to expand program protection to include system assurance. This guidebook only provides a recommended expansion of assurance requirements to the 5200.1-M baseline.
6. Countermeasures. Countermeasures include anything that effectively negates an adversary's ability to exploit weaknesses and vulnerabilities. Examples include physical protection of facilities, personnel clearance processes, hardening of development infrastructure (particularly for configuration management), element certifications, etc. This should be a time-phased plan of protection. See DoD 5200.1-M, C3.9, and guidebook 3.3.5, 3.3.6, 3.4.3, 3.4.4, and 3.4.5.
7. Cost. Discuss how much the protection will cost, compared to the cost of loss of assurance. The cost of loss of assurance includes the total *impact* from losing the missions that depend on the system, and the total cost of system development and deployment. See DoD 5200.1-M, C3.10, and guidebook 3.4.7.
8. Security Classification Guide. Develop guidance to identify information that should protected, and to what degree. For system assurance, this includes not just confidentiality, but also integrity, availability, authentication, accountability (including non-repudiation), and auditability. Ensure that system maintenance does not become an easy mechanism for subversion. This should be phased by time or event, and explain what to protect, for what reason. DoD 5200.1-M primarily focuses on classified elements. See DoD 5200.1-M, C4.2.1; See DoD 5200.1-M, C4; Sections 3.1.1, Acquisition, and 3.1.2, Supply Process.

9. Technology Assessment/Control Plan. Consider the feasibility of the program's participation in joint programs, considering the issues of disclosure and technical security perspective. Make decisions on foreign sales, commercial sales, and co-production or licensed production of the system, as well as on the feasibility of international cooperative agreements. See DoD 5200.1-M, C5; Sections 3.1.1, Acquisition, and 3.1.2, Supply Process).
10. System Security Engineering Approach. Identify engineering and design approaches that will be used to eliminate, reduce, or control characteristics that could result in the deployment of systems with operational security deficiencies. See DoD 5200.1-M, C6; guidebook 3.4.3–3.4.11.

4.3 DoD Life Cycle Framework

The following subsections describe the assurance activities associated with the various reviews in the entire Integrated Defense Acquisition, Technology, and Logistics Life Cycle Management Framework.

4.3.1 Concept Refinement Phase

"Pre-acquisition, beginning with Concept Refinement, presents the first substantial opportunity to influence systems design by balancing technology opportunities, schedule constraints, funding availability, performance parameters, and operational requirements. Desired user capabilities, expressed in terms of Key Performance Parameters (KPPs) and other parameters, should be defined in terms of:

- Quantifiable metrics (e.g., speed, lethality) of performance to meet mission requirements affordably; and
- The full range of operational requirements (reliability, maintainability, system health, effectiveness, logistics footprint (include spares/supply chain management), supportability, etc.) to sustain the mission over the long term" (DAG).

The operational requirements necessary to sustain the mission must include the top-level system assurance claims to counter identified threats to the mission. These may be stated as KPPs.

Examine the operational requirements to answer the following:

- What role does the system play in the Global Information Grid (GIG)? What services does it provide? What service does it consume?
- Who are the system's customers and what role do they play? What level of system assurance do they require before they would be willing to use it or provide information to it?
- Will the system defend itself against attacks received through the GIG? How?
- Will it avoid adding new vulnerabilities to the GIG? For example, does it create new network connectivity that is less well-protected than existing connectivity?
- Where does the system touch other systems, making it vulnerable or other systems vulnerable to it?
- Who are the single-source/key suppliers? Is there an assurance issue with them (e.g., could they be easily subverted)?
- What other systems are relied on for delivery of service?

- Is there a way to minimize/reduce the number of elements that need assurance?

Information assurance is part of the Net-Ready KPP as discussed in CJCSI 6212.01D, 3/8/2006. The NR-KPP along with other KPPs can be augmented for system assurance issues.

4.3.1.1 Initial Technical Review

"The Initial Technical Review (ITR) is a multi-disciplined technical review [that] ensures that a program's technical baseline is sufficiently rigorous to support a valid cost estimate (with acceptable cost risk), and enable an independent assessment of that estimate by cost, technical, and program management subject matter experts. The ITR assesses the capability needs and conceptual approach of a proposed program and verifies that the requisite research, development, test, engineering, logistics, and programmatic bases for the program reflect the complete spectrum of technical challenges and risks" (DAG).

For the ITR the threats to the mission and their mission impact need to be identified. Initial system MAC and CLs and initial top-level system assurance claims need to be defined to address them. The identified threats, mission impact, and claims are reviewed as part of the ITR by system assurance and information assurance subject matter experts, and are incorporated into the consideration of program technical challenges and risks.

System assurance need not cost more when considering the full system life cycle, but does require early consideration to be effectively and efficiently implemented. Examine the need for long-lead items associated with critical technologies, and their anticipated cost implications.

For more information, see Section 3.4.1, Stakeholder Requirements Definitions.

4.3.1.2 Alternative Systems Review

"The ASR [Alternative Systems Review] is a multi-disciplined technical review to ensure that the resulting set of requirements agrees with the customers' needs and expectations and that the system under review can proceed into the Technology Development phase… Generally this review assesses the alternative systems that have been evaluated during the Concept Refinement phase, and ensures that the preferred system alternative is cost effective, affordable, operationally effective, and suitable, and can be developed to provide a timely solution to a need at an acceptable level of risk. [This requires understanding of] operational effectiveness and technology risks…." (DAG).

To successfully complete the ASR review, ensure the following system assurance items were satisfactorily completed:

- System threats and system assurance claims were considered as part of the analysis of alternatives and full system life cycle costs were used in the analysis. Sometimes the seemingly cheapest alternative has higher system life cycle costs, due to assurance complications.
- A preliminary identification of critical technologies with a description of how to assure these technologies. This list will eventually feed the Critical Program Information (CPI) developed at the start of the System Development and Demonstration Phase. As an aid to this identification, review the Military Critical Technologies List (http://www.dtic.mil/mctl/).
- The Initial Capabilities Document (ICD) and Preliminary System Specification includes:

- The requirement for the development of an assurance case with high-level claims for each system determined to be critical.
- The sustaining mission operational requirements contain the top-level system assurance claims to counter identified threats to the mission. They should broadly identify an approach for developing the system assurance case.
- A critical elements list.
- The Support and Maintenance Concepts and Technologies with a description of how assurance will be maintained.

- Technical Issues/Technical Development Strategy/Market Research
  - If new high-assurance capabilities are needed, identify them as needs for technology development. This involves the need for (1) new approaches to implement high assurance, or (2) implementing high assurance in specific critical elements. High-assurance elements are often long-lead items.
  - Strategy should identify how to counter subversion via suppliers of critical new technology.
  - Ensure any necessary technology transfer limitations will be in place to prevent theft of DoD-unique critical technologies.
- Analysis of Alternatives (AoA)/Economic Analysis/Component Cost Analysis and Cost Analysis Requirements Description
  - Examine each alternative for feasibility of meeting system assurance needs.
  - Key interfaces and protection to/from them? How does it access the GIG (directly? satellite, wired, etc.)? How assurable will the selected alternative be (look at system breakdown, what is the strategy to assure each element)? For example, a GPS-guided missile risks GPS being jammed; an alternative is having a backup inertial system, so that jamming doesn't disable the system.
  - Analyze potential long-lead time items, or cost-drivers such as anti-tamper and high-assurance and MAC I requirements. These analyses will be used later in developing the PPP and information assurance strategy for the chosen alternative.
- Systems Engineering Plan (SEP)/Test and Evaluation Strategy (TES)/Test and Evaluation Master Plan (TEMP)
  - Ensure that the SEP and TES, as well as any draft TEMP, include the adoption of these system assurance guidelines or provide rationale for variance.
  - Some programs find it useful to merge some system assurance requirements into the System Safety Analysis and/or develop a parallel System Security Analysis.
  - These additions affect the Integrated Master Schedule (IMS).

For more information, see Section 3.4.1, Stakeholder Requirements Definition.

4.3.1.3 Milestone A

The Milestone A decision point approves a preferred system concept and the technology development strategy, and initiates the Technology Development phase. For more information on the DoD Instruction 5000.2 Information Requirements for Milestone A, see https://akss.dau.mil/dag/FWST/milestone-a.htm.

To successfully complete Milestone A, ensure the following system assurance items were satisfactorily completed:

- Examination of required critical system assurance capabilities to reduce the risk that the system will adversely impact the enterprise (DoD). It is DoD policy to manage all interconnections of DoD information systems "to continuously minimize community risk by ensuring that the assurance of one system is not undermined by vulnerabilities of interconnected systems." (DoDD 8500.1 section 4.14)
- Exit criteria for the rest of the system life cycle should include system assurance criteria, which should be captured in the Acquisition Decision Memorandum (ADM) before sign-off, and refined through the life cycle.

For more information, see Section 3.4.1, Stakeholder Requirements Definitions.

4.3.2 Technology Development Phase

Achieving the necessary system assurance may require planning for long-lead-time technologies as critical technology elements. For example, algorithms constituting critical technologies or critical functionality, and which will execute on ASICs, will be required to be developed in a trusted access program office (TAPO)-approved foundry. In these and other circumstances involving critical technology comprising CPI, advanced planning for anti-tamper should be investigated. Similarly, verifying software to very high levels of assurance requires investment in technology/tools to enable this. Presuming that the commercial world will do this on its own is naïve; while there are some commercial industries where assurance is critical (e.g., the financial industry), most commercial products are not required to withstand the attacks that defense systems must endure. Investing early in key R&D efforts, where necessary for system assurance, can provide assurance at a far lower cost and risk.

Disruptive technologies, not specifically focused on assurance, have assurance ramifications that may not be fully understood. They are often developed by organizations focused on just getting the technologies to work, rather than on assuring them. Moreover, they may be developed by people/organizations whose trustworthiness is unknown or questionable.

The Military Critical Technologies List can be leveraged as a resource for deciding on technology and its ability to meet system assurance requirements.

4.3.2.1 Acquisition Strategy

Throughout the technology development phase, acquisition strategy is developed.

The following criteria need to be included as part of the final acquisition strategy:

- Ensure that the Acquisition Strategy identifies the technical, schedule, cost, and funding issues associated with acquiring the appropriate levels of system assurance. The planning for and documentation of the Acquisition IA Strategy should produce the information required for this section.
- Address the required MAC and CL, when developing the Acquisition Strategy.
- Benefit Analysis and Determination (applicable to bundled acquisitions) (part of acquisition strategy)
    - Where assurance is important, prefer the development of small systems with narrow requirements because smaller systems tend to be easier to assure.

- Ensure that bundling does not adversely affect system assurance. Bundling may not be a cost savings, as the reduced cost for the functional requirements could be offset by the assurance efforts for a more complex system.

- Industrial Capabilities/Competition Analysis (part of acquisition strategy)
  - Ensure that system assurance is aided, and not adversely affected, by the supply strategy (including system elements and maintenance). See Section 4.2.8, Supplier Assurance.
- Core Logistics Analysis/Source of Repair Analysis (part of acquisition strategy)
  - Ensure that the logistics/source of repair are adequately protected from subversion (from both external and internal sources).
- Cooperative Opportunities (part of acquisition strategy)
  - When increased system assurance is being delivered, this should be noted as an aid to opportunities for cooperation. This is because increased system assurance reduces the risk of partners subverting other systems (intentionally or unintentionally).
- Other Acquisition Strategy
  - Open interface standards should be part of the acquisition strategy. Standard interfaces enable users to switch between products, if (1) the element's vulnerabilities are found to be unacceptable, or (2) a tool's ability to find vulnerabilities is unacceptable. This then provides a motivation for suppliers to improve their products, and for systems to use those improvements.
  - Best practices should be used, as noted in 8500.2 IA control DCBP-1.
- Program Protection Plan (includes Anti-Tamper Annex) (Acquisition Strategy/Research and Tech Protection)
  - Ensure that a preliminary CPI list has been developed (see DoDI 5200.39).
  - PPPs often only cover confidentiality of CPI. Ensure that the integrity, availability, and auditability of critical elements are also covered in the PPP.
  - Traditionally in the DoD, technologies are not protected until Milestone B is passed, and only for confidentiality (not revealing the technology to enemies). For assurance, protection needs to begin sooner, and also consideration should be given to integrity and availability of the technology.
  - CPI needs to be entered into the horizontal protection database.
  - Noncritical program technologies and elements need to be compared to the CPI horizontal database to determine if any of these technologies need to be added to the program's CPI because they are critical to another program.
- Exit criteria for the rest of the system life cycle should include system assurance criteria – possibly via reference to this guidebook

4.3.2.2 System Requirements Review

The System Requirements Review (SRR) is a multi-disciplined technical review to ensure that the system under review can proceed into the System Development and Demonstration phase, and that all system requirements and performance requirements derived from the ICD or draft Capability Development Document (CDD) are defined and are consistent with cost (program budget), schedule (program schedule), risk, and other system constraints (DAG).

System Requirements Reviews occur in multiple phases. An SRR in the Technology Development phase does not have all of the information that is available during an SRR in System Development and Demonstration, but, from the SA standpoint, the same approach should be used.

To successfully complete the SRR, ensure the following system assurance items were satisfactorily completed:

- ICD and Preliminary System Performance Specification
  - Establishes the requirement for the development of an assurance case, including high-level claims, for a system determined to be critical.
  - The operational requirements necessary to sustain the mission include the top-level system assurance claims that address identified threats to the mission that are the foundation for the assurance case. Critical elements are identified.
  - The Support and Maintenance Concepts and Technologies documented with a description of how assurance will be maintained.
  - Requirements with system assurance implication have been tagged for SA traceability and verification.
- Identification of all critical elements to be protected, and what aspects of them are to be protected (e.g., confidentiality, integrity, availability, authentication, accountability (including non-repudiation), and auditability). For example, ensure that an adversary cannot gain control over a weapon system.
  - Initial identification of potential CPI, and a preliminary approach to the protection of that CPI is part of the system requirements.
- Identification of all relevant system assurance threats and their potential impact on critical system assets.
- Identification of high-level potential weaknesses in the system (where practicable)
- Determination and derivation of system assurance requirements (as a subset of the system requirements).
- Determination and derivation of system assurance claims and incorporation of the claims into the system assurance requirements.
- For each system assurance claim, determination of the general approach to justify the claim. The justification must be sufficient to provide confidence that the assets are protected against the threats in spite of weaknesses.
- Creation of a mapping from the system assurance requirements to the assets, threats, and weaknesses, to ensure that the requirements cover them. (This provides traceability to justify that these requirements are necessary and sufficient for assurance.)
- Selection of the system assurance requirements (or their portions) that are critical.
- Derivation of the system assurance claims from the critical system assurance requirements.
- Analysis of Alternatives/Affordability Assessment/Economic Analysis/Component Cost Analysis/Cost Analysis Requirements Description
  - Examination of each alternative for feasibility of meeting system assurance needs.
  - Refine identification of CPI for the alternatives, and consider the effort to protect them as part of trade-off.

- Ensure that cost-reduction measures do not produce unacceptable assurance limitations.

- Risk Assessment
  - Inclusion of the risks of intentional and unintentional vulnerabilities in elements, both from external suppliers and custom elements, with a description of how to mitigate.
  - Assessment of assurance risks.
- System Threat Assessment/Capstone Information Operations System Threat Assessment includes:
  - Unintentional or intentional vulnerabilities in supplied elements.
  - Unintentional or intentional vulnerabilities in developed elements and final integrated system.
  - Unintentional or intentional vulnerabilities in the system during operation.
  - Probability of attempted exploitation, including who and how[2].
  - Supplier assurance; see Section 4.2.8, Supplier Assurance.
  - System threat assessment; see DoD Directive 5105.21 for additional information on how to address system threat assessment matters.
  - Ensure that DAA is involved, and that they have accepted the proposed countermeasures for such threats.
  - Analysis of potential threats within all areas of DOTMLPF(Doctrine, Training, Material, Leadership, Personnel, and Facility).
- NR-KPP Certification and Joint Interoperability Test Command (JITC) interoperability certification
  - NR-KPP certification in some cases is required for systems. Ensure that the certification includes system assurance considerations.
  - The JITC interoperability certification feeds into the NR-KPP certification and thus provides evidence in support of the certification.
- Systems Engineering Plan (SEP)
  - Examine the SEP to ensure that system assurance is covered throughout the life cycle. More specifically, the assurance claims for the system and any associated plan for providing evidence and arguments should be developed. See the following system development and demonstration subsections for the types of activities that should be noted.
  - Use the "Systems Engineering Plan Preparation Guide" http://www.acq.osd.mil/sse/docs/SEP-Prep-Guide-Ver-2-0_18Oct07.pdf. Refer to Section 3, General System Assurance Guidance (especially Section 3.4, Technical Processes) for assurance-specific additions to the SEP guidance.
  - Verify that the additional assurance activities are incorporated into the IMS.
- Test and Evaluation Master Plan (TEMP)

[2] The probability of an attack is impacted by both the capability of a threat agent to do harm and the intent of that agent to do harm (see Joint Chiefs of Staff Publication 1 (JCS Pub 1)).

- Examine the TEMP to ensure testing processes are sufficient for system assurance. This requires that those system assurance claims supported by testing will be adequately supported by appropriate test results.
- See general system assurance guidance in Sections 2 and 3.

- System Support and Maintenance Objectives and Requirements
  - Includes requirements to maintain assurance; see Section 3.4.8, Validation.
- Independent Cost Estimate (Cost Analysis Improvement Group (CAIG)) and Manpower Estimate
  - Ensure that adequate system assurance efforts are included in the estimates.
- Development of a traceability matrix mapping of applicable DODI 8500.2 controls to requirements baseline
- Incorporation of applicable DODI 8500.2 IA control requirements into the requirements baseline. For example:
  - Incorporation of a requirement for best security practices such as SSO, PKE, smart card, and biometrics (see IA control: DCBP-1).
  - Incorporation of a requirement for protection of external interfaces, user roles, unique security requirements (encryption of key data elements at rest), HIPAA, privacy act, and Sarbanes-Oxley (see IA control: DCFA-1).
  - Incorporation of a requirement to ensure that the system meets IA accreditation requirements (DCII-1).
  - Incorporation of a requirement to define all IA roles and responsibilities in writing with criteria such as clearance and training requirements (DCSD-1, PRTN-1).
- Ensure the planning for an Integrated Product Team (IPT) includes SA expertise.

For more guidance, see Sections 3.4.1, Stakeholder Requirements Definitions, and Section 3.4.2, Requirements Analysis.

4.3.2.3 Integrated Baseline Review

Ensure that:

- Technical scope of work accurately addresses assurance issues consistent with authorizing documents.
- Sufficient resources (budgets, facilities, personnel, skills, etc.) are assigned to develop the assurance claims, arguments, and evidence.

For more information, see Section 3.4.1, Stakeholder Requirements Definitions.

4.3.2.4 Technology Readiness Assessment

"The TRA [Technology Readiness Assessment] is a systematic, metrics-based process that assesses the maturity of Critical Technology Elements. The TRA should be conducted concurrently with other Technical Reviews, specifically the Alternative Systems Review, System Requirements Review, or the Production Readiness Review. If a platform or system depends on specific technologies to meet system operational threshold requirements in development, production, and operation, and if the technology or its application is either new or novel, then that technology is considered a Critical Technology Element… If the system does not meet pre-defined Technology Readiness Level (TRL) scores, then a Critical Technology Element

maturation plan is identified. This plan explains in detail how the Technology Readiness Level will be reached prior to the next milestone decision date or relevant decision point" (DODI 5000.2),

The following situations require maturation plans to achieve the necessary assurance:

- An element that is functionally mature, but immature with respect to the assurance requirements of the system, requires a maturation plan when it is a critical element.
- The infrastructure necessary to create an element in an assured manner may not be available. For example, fabrication of trusted ICs with certain properties is not feasible using current equipment/infrastructure (e.g., trusted foundries are not able to produce those ICs).
- There may be no mature process available for achieving the required assurance. For example, high-assurance verification of software is difficult and has traditionally not scaled well, so building large high-assurance software requires additional investment in process technology.

To successfully complete the TRA review, ensure the following system assurance items were satisfactorily completed:

- Consideration of Technology Issues/Market Research/TRA
  - Identify the most promising assurance tools and techniques for use in system development and demonstration, and ensure that they are sufficiently mature.
  - If new high-assurance capabilities are needed (as identified in milestone A), ensure that they have been adequately matured for use in system development and demonstration. This includes (1) new approaches to implement high assurance, or (2) implementing high assurance in specific critical elements. Developing high assurance elements is often a long-lead item.
- Technology Development Strategy
  - See the discussion in Milestone A
- Technology Readiness Assessment
- Independent Technology Assessment (Acquisition Category (ACAT) ID only)

These factors must be considered as part of a TRA. For more information, see the Architectural Design Process section (3.4.3) and the Agreement Process section (3.1) of this document.

4.3.2.5 Milestone B

Milestone B confirms that technology risk has been reduced and determines that the appropriate set of technologies will be integrated into a full system. The Technology Development Strategy (TDS) and the draft CDD have led to a baselined CDD. The System Performance Specification and the AoA are delivered. While the RFP for the system development and demonstration phase can be issued prior to Milestone B approval, the contract cannot be awarded until Milestone B is completed. For more information on the DoD Instruction 5000.2 Information Requirements for Milestone B, see https://akss.dau.mil/dag/FWST/milestone-b.htm.

To successfully complete Milestone B, ensure the following system assurance items were satisfactorily completed:

- Confirm that SRR, IBR, and TRAs were successfully completed including assurance criteria noted above. This includes review of the acquisition strategy and the activities supporting it
- Ensure that an acquisition strategy has been completed, including the issues noted above.

For more information, see Sections 3.4.1, Stakeholder Requirements Definitions, and 3.4.2, Requirements Analysis.

4.3.3 System Development and Demonstration Phase

"In System Development and Demonstration, the program, the system architecture, and system elements down to the configuration item level are defined based upon the mature technology suite selected and integrated during Concept Refinement and Technology Development" (DAG). Also, "No contract for the acquisition of a Mission Critical or Mission Essential IT system may be awarded until the DoD CIO has determined that there is in place for the system an appropriate information assurance strategy" (DAG 7.8.2).

This phase may also include an integrated baseline review (IBR) and technical readiness assessment (TRA). For more information on IBR and TRA with respect to assurance, see Section 4.3.2, Technology Development Phase.

Some deliverables that require new system assurance material include the SEP, TRA, TEMP, the System Threat Assessment, and the Risk Assessment. Intermediate products, such as the system requirements and architecture, should be delivered and maintained as part of the outputs of system development, so that they can be used in later system maintenance. This helps provide the traceability to maintain the system's assurance.

4.3.3.1 System Requirements Review

"The SRR [System Requirements Review] is a multi-functional technical review to ensure that all system and performance requirements derived from the capability development document (CDD) are defined and consistent with cost (program budget), schedule (program schedule), risk, and other system constraints" (DAG).

To successfully complete the SRR, ensure the following system assurance items were satisfactorily completed:

- CDD and System Performance Specification
  - Establishes the requirement for the development of an assurance case, including high-level claims, for a system determined to be critical.
  - The operational requirements necessary to sustain the mission include the top-level system assurance claims that address identified threats to the mission. It should broadly identify an approach for developing the system assurance case.
  - Critical elements are identified.
- Identification of all critical elements to be protected, and what aspects of them are to be protected (e.g., confidentiality, integrity, availability, non-repudiation, authentication, and/or audit). For example, ensure that an adversary cannot gain control over a weapon system.
  - Identification of CPI, and ensuring that CPI protection (including anti-tamper considerations) is part of the system requirements.

- Identification of all relevant system assurance threats and their potential impact on critical system assets.
- Identification of high-level potential weaknesses in the system (where practicable).
- Determination and derivation of system assurance requirements (as a subset of the system requirements).
- Determination and derivation of system assurance claims and incorporation of the claims into the system assurance requirements.
- For each system assurance claim, determination of the general approach to justify the claim. The justification must be sufficient to provide confidence that the assets are protected against the threats in spite of weaknesses.
- Creation of a mapping from the system assurance requirements to the assets, threats, and weaknesses, to ensure that the requirements cover them. (This provides traceability to justify that these requirements are necessary and sufficient for assurance.)
- Selection of the system assurance requirements (or their portions) that are critical.
- Derivation of the claims necessary for the assurance case from the critical system assurance requirements.
- Risk Assessment
  - Include the risks of intentional and unintentional vulnerabilities in elements, both from external suppliers and custom elements, with a description of how to mitigate.
  - Assessment of assurance risks
- System Threat Assessment/Capstone Information Operations System Threat Assessment include:
  - Unintentional or intentional vulnerabilities in supplied elements
  - Unintentional or intentional vulnerabilities in developed elements and final integrated system
  - Unintentional or intentional vulnerabilities in the system during operation
  - Probability of attempted exploitation, including who and how
  - Supplier assurance; see Section 4.2.8, Supplier Assurance.
  - System threat assessment; see DoD Directive 5105.21 for additional information on how to address system threat assessment matters
  - Analysis of potential threats within all areas of DOTMLPF (Doctrine, Training, Material, Leadership, Personnel and Facility)
- Systems Engineering Plan (SEP)
  - Examine the SEP to ensure that system assurance is covered throughout the life cycle. More specifically, the assurance claims for the system and any associated plan for providing evidence and arguments should be developed. See the following system development and demonstration subsections for the types of activities that should be noted.
  - Use the “Systems Engineering Plan Preparation Guide” http://www.acq.osd.mil/sse/docs/SEP-Prep-Guide-Ver-2-0_18Oct07.pdf. Refer back to Section 3 (especially the technical processes section) of this document for assurance-specific additions to the SEP guidance.
  - Verify that the additional assurance activities are incorporated into the IMS.
- Test and Evaluation Master Plan (TEMP)

- Examine the TEMP to ensure testing processes are sufficient for system assurance. This requires that those system assurance claims supported by testing will be adequately supported by appropriate test results.
- See also Sections 2 and 3.

- System Support and Maintenance Objectives and Requirements
  - Includes requirements to maintain assurance; see Section 3.4.8, Validation.
- Development of a traceability matrix mapping of applicable DODI 8500.2 controls to requirements baseline
- Incorporation of all applicable DODI 8500.2 IA control requirements into the requirements baseline. For example:
  - Incorporation of a requirement for best security practices such as SSO, PKE, smart card, and biometrics (see IA control: DCBP-1).
  - Incorporation of a requirement for protection of external interfaces, user roles, unique security requirements (encryption of key data elements at rest), HIPAA, privacy act, and Sarbanes-Oxley (see IA control: DCFA-1).
  - Incorporation of a requirement to ensure that the system meets IA accreditation requirements (DCII-1).
  - Incorporation of a requirement to define all IA roles and responsibilities in writing with criteria such as clearance and training requirements (DCSD-1, PRTN-1).

For more guidance, see Sections 3.4.1, Stakeholder Requirements Definitions, and 3.4.2, Requirements Analysis.

4.3.3.2 System Functional Review

"The SFR [System Functional Review] is a multi-functional technical review to ensure that the system under review can proceed into preliminary design, and that all system requirements and functional performance requirements derived from the capability development document (CDD) are defined and consistent with cost (program budget), schedule (program schedule), risk, and other system constraints" (DAG).

To successfully complete the SFR, ensure the following system assurance items were satisfactorily completed:

- Refinement of the system assurance requirements, including the definition of system assurance critical scenarios.
- Update of the system assurance case, based on additional information on assets, threats, and weaknesses.
- For each system assurance claim, determine the specific approach to justify the claim.
- Determination of the potential defensive mechanisms and their derived requirements, sufficient for the system assurance claims; ensure these functions are captured in the system's Functional Baseline. The system's verification planning should be updated and included as part of the Functional Baseline.
- Determine performance impacts (including reliability, maintainability, testability (system health) availability, safety, affordability, throughput, response time, etc.) of the system assurance requirements, if any, and reconcile if necessary.
- Where specific elements are known or dictated, system assurance requirements for them are allocated, and at least one workable assurance approach has been identified.

- Examination of analyses of alternatives to determine that the chosen alternative is feasible to assure. Beware of cost-reduction measures that produce unacceptable assurance limitations.
- Identification of the assurance requirements on development infrastructure (e.g., program language selection, development frameworks, middleware selection).
- Determination of how the system's security posture will be maintained throughout its life cycle (including periodic re-accreditation).

To complete the review, ensure that these were satisfactorily accomplished as exit criteria. For more guidance, see Sections 3.4.1, Stakeholder Requirements Definitions and 3.4.2, Requirements Analysis.

4.3.3.3 Preliminary Design Review

"The PDR [Preliminary Design Review] is a multi-disciplined technical review to ensure that the system under review can proceed into detailed design, and can meet the stated performance requirements within cost (program budget), schedule (program schedule), risk, and other system constraints" (DAG).

To successfully complete the PDR, ensure the following system assurance items were satisfactorily completed:

- Use the architecture and preliminary design information (as available) to identify critical components. Identify weaknesses and their associated potential vulnerabilities. Note that a weak architecture can result in a systemic weakness, which can in turn lead to many vulnerabilities. Thus a rigorous review of the architecture may be required prior to release to design phase. Refine and document a baseline of attack scenarios of the identified threats and assets (see IA control: VIVM-1).
- Development of specific instances of system assurance scenarios, at least for the critical system assurance requirements, to verify that the system will counter the attack.
- Architecture and preliminary system design includes IA accreditation requirements in its relationship to all hosting enclaves and impact analysis is completed for the architecture. Develop a list of all hosting enclaves as a baseline for tracking purposes as the system moves into the detailed design phase (see IA controls: DCII-1, DCID-1)).
  - Ensure that the architecture and preliminary system design of mobile code usage is evaluated for acceptable risk, avoiding high risk as defined by DOD requirements (see IA controls: DCMC-1 (Mobile Code)).
  - Exclude binary or machine executable public domain software products with no warranty and no source code (see IA control: DCPD-1).
- Ensure the validation plan includes methods that are focused on minimizing flaws or malformed software that can impact integrity or availability (e.g., buffer overruns, cross-site scripting issues) (see IA controls; DCSQ-1 (software Quality)).
- Use system assurance requirements and the instances of system assurance scenarios as part of the lower-level design trade-off analysis, and document the results.
  - Examine analyses of sub-element alternatives to determine that the chosen sub-element best meets requirements and fulfills the scenarios.
  - Consider both intrinsic and defensive functions.
  - Beware of cost-reduction measures that produce unacceptable assurance limitations.

- Examine defensive functions to reduce risks from other elements and provide evidence for assurance case.
- Ensure that the system design incorporates the following security, partitioning, access, and traceability mechanisms which are key defensive functions to assure the system:
  - that protection mechanisms with each external interface and associated security requirements are identified and incorporated into the architecture and preliminary design phases (See IA control: DCFA-1, ECIC-1).
  - that the system's preliminary design incorporates best security practices such as identification/authentication (individual and group) using DOD PKI, logon including single sign-on, PKE, key management, smart card, and biometrics (see IA controls: as per DoDI 8500.2 DCBP-1, IATS-2, IAAC-1, IAGA-1, IAIA-2, IAKM-3, ECLO-2, Best Security Practices).
  - that the architecture and design of the system ensures that data, including classified data, are accessed, changed, and logged only by authorized personnel with appropriate "need-to-know" (see IA control: ECCD-2, ECAN-1).
  - that user interface services for logical or physical separation are architected and designed to provide appropriate protections for data storage and management services. This is particularly important in high assurance systems (see IA controls: DCPA-1).
  - that security support structure is isolated by means of partitions, domains, etc including control of access. Examples include sandboxes, jails, and separation kernels (see IA controls: DCSP-1).
- Ensure that a comprehensive design for confidentiality, integrity availability, and non-repudiation is incorporated into the system (see IA controls: DCNR-1 (non-repudiation), ECCR-2 (confidentiality for data at rest), ECCT-2 (confidentiality for data in transit), ECWN-1 (confidentiality for wireless)).
- Ensure that the system architecture that enables DoD information systems to store, process, transmit, or display data is appropriately marked, labeled, and encrypted (see IA control ECML-1, ECNK-1, ECNK-2).
- Access control mechanisms such as the following must be implemented:
  - Ensure least privilege is incorporated into the system architecture and preliminary design (see IA control: ECLP-1).
  - Remote access for privileged functions is permitted for completing operational needs and is strictly controlled, and remote access to user functions is mediated through a managed access control point (see IA controls; EBRP-1, EBRU-1).
- The architecture and design should include audit mechanisms that deliver automated continuous on-line audit trail capability (see IA controls: ECAT-2, ECRG-1, ECTP-1, ECLC-1, ECAR-3).
  - Protection of audit trail from unauthorized access, modification, or deletion.
  - Tools for monitoring, analysis, review, and report generation of audit records.
  - Automatic recording, creation, deletion, or modification of security labels (where appropriate).
- The preliminary design should consider the events that can trigger system initialization, shutdown, abort, and recovery, as well as the impact of such activities on the system. (see IA control: DCSS-2, COTR-1).

- Completion of an Information Security technology evaluation of all critical COTS/GOTS elements as part of the analysis of alternatives (See IA control: DCAS-1, ECTM-2).
  - Examples of evaluations include certifications (e.g. Common Criteria) and verified implementation of integrity mechanisms.
  - High-robustness COTS/GOTS are assessed during the architecture development phase (Analysis of Alternatives – AoA) where required, e.g., when information transits networks at lower classification levels (see IA control: DCSR-3) and its processes have been performed.
- Updated the system requirements, the functional baseline and the allocated baseline to incorporate the claims, arguments, and scenarios,
- Captured assurance designs in the associated configuration item build-to documentation as part of the system's Product Baseline. The system's configuration item verification planning should be updated and included as part of the Product Baseline.,
- Updated system assurance case based on the design, new weaknesses, vulnerabilities identified and the preceding analysis.
  - For each system assurance claim, define the detailed argument(s) to be used to justify the claim, identify the expected evidence (type and expected measure) that will support the argument, and how that evidence will be acquired (including what verification data must be created to acquire that evidence).
  - After CDR, the assurance case's claims and argument structure are baselined, some evidence is already available, and the methods for acquiring the remaining evidence have been defined.
- Defined and selected assurance-specific static analysis and assurance-specific criteria to be examined during peer reviews, to be performed during implementation.
  - This should be documented in the SEP and Software Development Plan (SDP).
  - Plan for training for assurance-unique static analysis tools and peer reviews.
  - Ensure that another party (such as a peer) will independently perform static analysis and test, and that the element being reviewed will be the element that will be delivered. This counteracts the risk of a developer intentionally subverting analysis and test, as well as aiding against unintentional errors.
- Updated system assurance risk assessment incorporated into the risk management plan.
- For Assurance, the preliminary configuration management plan must support traceability and protection of each configuration item, including requirements and architectural elements (See IA Controls: DCPR-1, PESL-1). Examples include limiting privileges to changing production code and data as well as preventing unauthorized access.
- Perform specialized analysis on high-level architecture to identify derived system assurance requirements and behavior. Examples of analyses that are used include FMEA/FMECA, FTA, Reliability Centered maintenance (RCM), RAM allocations/predictions, R&M block diagram. RAM Program Plan, System Safety Program Plan and analysis for anti-tamper protection.
- Updated the system assurance case based on additional information on assets, threats, and weaknesses. Operational Test & Evaluation (OT&E) reports on previous versions, if available, may provide useful information on weaknesses. Particularly relevant are any information assurance assessments on the previous version's abilities to protect, detect, react/respond, and restore.

- Used system assurance requirements and critical system-assurance scenarios as part of the system-level architectural trade-off analysis (analysis of alternatives), and document the results.
  - Examine analyses of alternatives to determine that the chosen alternative is feasible to assure both intrinsic and defensive functions. In some cases, defensive functions, which reduce risks from other elements and provide evidence for assurance case.
  - For all critical elements being considered for procurement, an analysis of the supplier and its processes has been performed.
  - Beware of cost-reduction measures that produce unacceptable assurance limitations.
- Updated the system assurance requirements, including derived requirements based on architectural analysis, and allocated the system assurance requirements to the architecture baseline (incorporate into the allocated baseline).
  - System assurance requirements have an emergent property from the set of subsystems and therefore each architectural change requires a reexamination of the architecture to determine whether the architecture still meets the system assurance requirements.
  - Each assurance requirement needs to be allocated to one or more architectural elements.
  - Alternatively, sometimes an additional defensive architectural element can be identified or developed so that an assurance requirement can be allocated solely to that architectural element.
  - Capture allocated assurance requirements and physical and functional architecture implications in the preliminary design specifications for each configuration item as part of the system's Allocated Baseline. The system's configuration item verification planning should be updated and included as part of the Allocated Baseline.
  - For interfaces, identify the filtering restrictions that the elements must enforce; e.g., for data interfaces, identify the legal values (so that all illegal values will be rejected/trapped); for electrical connections, identify where voltage limits must be enforced.
- For each system assurance claim, define the overall argument(s) to be used to justify the claim, and identify the expected evidence (type and expected measure) that will support the argument
- Identification of weaknesses and vulnerabilities associated with intended development environment/infrastructure, including the need for additional tools and guidelines to implement system assurance requirements.

For more detailed guidance, see Section 3.4.3, Architectural Design.

4.3.3.4 Critical Design Review

"The CDR [Critical Design Review] is a multi-disciplined technical review to ensure that the system under review can proceed into system fabrication, demonstration, and test, and can meet the stated performance requirements within cost (program budget), schedule (program schedule), risk, and other system constraints" (DAG).

To successfully complete the CDR review, ensure the following system assurance items were satisfactorily completed:

- Used detailed design information to identify weaknesses and their associated vulnerabilities. Refined and baselined attack scenarios of the identified threats and assets. (See IA control: VIVM-1).
- Ensured that the Final Anti-Tamper Plan has been approved by the program manager and has received Service concurrence prior to the CDR.
- Selected additional development tools and guidelines to counter weaknesses and vulnerabilities in the system elements and development environment(s). These include static analysis tools, etc.
- Developed specific instances of system assurance scenarios, at least for the critical system assurance requirements, to verify that the system will counter the attack.
- Updated specialized analysis on design to identify derived system architecture and behavior changes. Examples include FMECA, FTA, RCM, and analysis for anti-tamper protection.
- Ensure design meets IA accreditation requirements (see IA controls: DCII-1, DCID-1)).
  - Developed and maintained a listing of hosting enclaves.
  - Evaluated mobile code usage for acceptable risk, avoiding high risk as defined by DoD and that various mechanisms are used to verify implementation to support assurance needs (see IA controls: DCMC-1).
  - Excluded binary or machine-executable public domain software products with no warranty and no source code (see IA control: DCPD-1).
- Ensured validation includes methods that are focused on minimizing flaws or malformed software that can impact integrity or availability (e.g. buffer overruns, cross-site scripting issues) (see IA controls; DCSQ-1).
- Used system assurance requirements and the instances of system assurance scenarios as part of the lower-level design trade-off analysis, and documented the results.
  - Examined analyses of alternatives to determine that the chosen alternative is feasible to assure both intrinsic and defensive functions.
  - Ensure cost-reduction measures have not produced unacceptable assurance limitations.
- Ensured that the system design incorporates the following security, partitioning, access, and traceability mechanisms which are key defensive functions to assure the system:
  - Protection mechanisms with each external interface and associated security requirements (See IA control: DCFA-1)
  - Best security practices such as identification/authentication (individual and group) using DoD PKI, logon including single sign-on, PKE, key management, smart card, and biometrics (See IA controls: DCBP-1, IATS-2, IAAC-1, IAGA-1, IAIA-2, IAKM-3, ECLO-2).
  - User interface services are logically or physically separated from data storage and management services. This is particularly important in high assurance systems (See IA controls: DCPA-1).
  - Security support structure is isolated by means of partitions, domains, etc including control of access. Examples include sandboxes, jails, and separation kernels (See IA controls: DCSP-1).
  - Confidentiality, integrity availability, and non-repudiation (See IA controls: DCNR-1, ECCR-2, ECCT-2, ECWN-1).

        - Information and DoD information systems that store, process, transmit, or display data are appropriately marked, labeled, and encrypted (See IA control ECML-1, ECNK-1, ECNK-2).
    - Access control mechanisms such as the following have been implemented
        - Least privilege (See IA control: ECLP-1)
        - Ensure data, including classified data, is accessed, changed, and logged only by authorized personnel with appropriate 'need-to-know' (See IA control: ECCD-2, ECAN-1).
        - Remote access for privileged functions is permitted for completing operational needs and is strictly controlled, and remote access to user functions is mediated through a managed access control point (See IA controls; EBRP-1, EBRU-1).
    - All authorizations to information contained within an object are revoked prior to initial assignment, allocation or reallocation (See IA control ECRC-1).
    - An automated continuous on-line audit trail creation capability that provides: (see IA controls: ECAT-2, ECRG-1, ECTP-1, ECLC-1, ECAR-3)
        - Protection of audit trail from unauthorized access, modification, or deletion
        - Tools for monitoring, analysis, review, and report generation of audit records
        - Automatic recording , creation, deletion, or modification of security labels (where appropriate)
    - System initialization, shutdown, aborts, and recovery are configured to insure the system remains in a secure and verifiable state (see IA control: DCSS-2, COTR-1).
    - An Information Security technology evaluation of all critical COTS/GOTS elements (See IA control: DCAS-1, ECTM-2).
        - Examples of evaluations include certifications (e.g. Common Criteria) and verified implementation of integrity mechanisms.
        - High robustness COTS/GOTS are used where required, e.g., when information transits networks at lower classification levels (see IA control: DCSR-3).
- Updated the system requirements, the functional baseline, and the allocated baseline to incorporate the claims, arguments, and scenarios.
- Captured assurance designs in the associated configuration item build-to documentation as part of the system's Product Baseline. The system's configuration item verification planning should be updated and included as part of the Product Baseline.
- Updated the system assurance case based on the design, new weaknesses, vulnerabilities identified, and the preceding analysis.
    - For each system assurance claim, define the detailed argument(s) to be used to justify the claim, identify the expected evidence (type and expected measure) that will support the argument, and how that evidence will be acquired (including what verification data must be created to acquire that evidence).
    - After CDR, the assurance case's claims and argument structure are baselined, some evidence is already available, and the methods for acquiring the remaining evidence have been defined.
- Defined and selected assurance-specific static analysis and assurance-specific criteria to be examined during peer reviews, to be performed during implementation.
    - Documented in the SEP and Software Development Plan (SDP) the analysis and peer review criteria.

- Planned for training for assurance-unique static analysis tools and peer reviews.
- Ensure that another party (such as a peer) will independently perform static analysis and test, and that the element being reviewed will be the element that will be delivered. This counteracts the risk of a developer intentionally subverting analysis and test, as well as aiding against unintentional errors.

- Updated system assurance risk assessment incorporated into the risk management plan.
- Supported assurance, traceability and protection of each configuration item through the configuration management (see IA Controls: DCPR-1, DCSL-1, ECPC-2, ECSD-2, ECVP-1, PESL-1).
  - Examples include limiting privileges to changing production code and data as well as preventing unauthorized access.

For more detailed guidance, see Section 3.4.3, Architectural Design.

4.3.3.5 Implementation and integration

Although there is no review process within the DAG for implementation and integration, it is important to note that there are significant assurance activities during implementation and integration. For more detailed guidance, see Sections 3.4.4, Implementation, and 3.4.5, Integration.

4.3.3.6 Test Readiness Review

"The TRR [Test Readiness Review] is a multi-disciplined technical review to ensure that the subsystem or system under review is ready to proceed into formal test" (DAG).

To successfully complete the TRR, ensure the following system assurance items were satisfactorily completed (Note: In some cases, the JITC interoperability certification process will leverage this information):

- Verification that (1) assurance-specific static analysis and (2) peer reviews of assurance criteria have been completed. Verification that another party (such as a peer) performed static analysis and peer review. Ensure that detected defects have been corrected or mitigated, and appropriately documented.
- Updated the assurance case with the relevant evidence.
- Selected additional test tools and developed guidelines to identify or verify weaknesses and vulnerabilities in the system elements and development environment(s). These include static analysis tools, etc.
- Developed (test input) data to be used in tests of system assurance requirements, based on the specific instances of system assurance scenarios.
- Refined the baselined attach/threat scenarios which, along with system implementation information, will be used to test for weaknesses and their associated vulnerabilities. This is also when the weaknesses and vulnerabilities are appropriately documented (see IA control: VIVM-1).
- Updated specialized analysis in system development to identify derived system architecture and behavior changes. Examples include FMECA, FTA, RCM, and analysis for anti-tamper protection.
- Ensured the developed system meets IA accreditation requirements (see IA controls: DCII-1). Define the expected results that will be satisfactory evidence to complete the

assurance case and meet the system assurance requirement. Ensure that the system incorporates the following security, partitioning, access, and traceability mechanisms that are key defensive functions to assure the system. The testing process should yield results which can then be used to derive evidence necessary for developing an robust assurance case:

- Implementation of protection mechanisms for each external interface and associated security requirements (see IA control: DCFA-1).
- Incorporation in the system of the best security practices such as identification/authentication (individual and group) using DoD PKI, logon including single sign-on, PKE, key management, smart card, and biometrics (see IA controls: as per DoDI 8500.2 DCBP-1, IATS-2, IAAC-1, IAGA-1, IAIA-2, IAKM-3, ECLO-2, ECPA-1, Best Security Practices).
- Separation of user interface services both logically or physically from data storage and management services. This is particularly important in high-assurance systems (see IA controls: DCPA-1).
- Security support structure is isolated by means of partitions, domains, etc., including control of access. Examples include sandboxes, jails, and separation kernels (see IA controls: DCSP-1).
- Ensure the system has been developed to achieve confidentiality, integrity, availability and non-repudiation (see IA controls: DCNR-1 (non-repudiation), ECCR-2 (confidentiality for data at rest), ECCT-2 (confidentiality for data in transit), ECWN-1 (confidentiality for wireless implementations in accordance with DOD policy)).
- Information and DoD information systems that store, process, transmit, or display data are appropriately marked, labeled, and encrypted (See IA control ECML-1, ECNK-1, ECNK-2).
- Access control mechanisms such as the following have been implemented, and appropriate test cases are generated to verify the implementation:
  - Least privilege (see IA control: ECLP-1).
  - Ensure data, including classified data, is accessed, changed, and logged only by authorized personnel with appropriate ‘need-to-know’ (See IA control: ECCD-2, ECAN-1).
  - Remote access for privileged functions is permitted for completing operational needs and is strictly controlled, and remote access to user functions is mediated through a managed access control point. (See IA controls; EBRP-1, EBRU-1).
  - All authorizations to information contained within an object are revoked prior to initial assignment, allocation, or reallocation (See IA control ECRC-1).
- Automated continuous on-line audit trail mechanisms are created/leveraged in the system implementation to provide the following (see IA controls: ECAT-2, ECRG-1, ECTP-1, ECLC-1, ECAR-3):
  - Protection of audit trail from unauthorized access, modification, or deletion.
  - Incorporation of tools for monitoring, analysis, review, and report generation of audit records.
  - Automatic recording , creation, deletion, or modification of security labels (where appropriate).

- System initialization, shutdown, aborts, and recovery are configured to ensure the system remains in a secure and verifiable state (see IA control: DCSS-2, COTR-1).
- An Information Security technology evaluation is completed and verifiable for all critical COTS/GOTS elements (see IA control: DCAS-1, ECTM-2).
    - ♦ Examples of evaluations include certifications (e.g., Common Criteria) and verified implementation of integrity mechanisms.
    - ♦ High-robustness COTS/GOTS are used where required, e.g., when information transits networks at lower classification levels (see IA control: DCSR-3) and its processes has been performed.

Configuration management supports assurance traceability and protection of each configuration item (see IA Controls: DCPR-1, DCSL-1, ECPC-2, ECSD-2, ECVP-1, PESL-1). Examples include limiting privileges to changing production code and data as well as preventing unauthorized access.

- Developed and maintained a listing of hosting enclaves. This includes the collection of evidence of deployment planning and coordination as well as the exchange of connection rules and requirements (see IA control: DCID-1).
- Verification that integrated mobile code has been evaluated for acceptable risk, avoiding high risk as defined by DoD. Assess the various mechanisms used to verify implementation to support assurance needs (see IA controls: DCMC-1). .
- Identification of industry tools and test cases to be used for the testing of any binary or machine-executable public domain software products with no warranty and no source code being used in the system (see IA control: DCPD-1).
- Ensure that static analysis has been performed (both source and binary) to identify weaknesses and vulnerabilities such as buffer overruns and cross-site scripting issues. Evaluate and incorporate the results of static analysis testing as evidence into the assurance case (see IA controls; DCSQ-1).
- Ensure that the system is implemented in compliance with ports, protocols, and services guidance according to the DoD information system requirements. The implementation should take into consideration both the development and configuration of the system to support appropriate integration as well as lock-downs (see IA control: DCPP-1 as well as the Defense Information Systems Agency (DISA) STIGS (http://iase.disa.mil/stigs/stig/index.html)).
- See section 4.3.3.7 on System Verification Review (SVR) and Production Readiness Review (PRR).

To complete the review, ensure that these were satisfactorily accomplished as exit criteria. For more detailed guidance, see the Verification Process section (3.4.6) of this document.

4.3.3.7 System Verification Review and Production Readiness Review

"The SVR [System Verification Review] is a multi-disciplined product and process assessment to ensure that the system under review can proceed into low-rate initial production (LRIP) and full-rate production (FRP) within cost (program budget), schedule (program schedule), risk, and other system constraints. The PRR [Production Readiness Review] examines a program to determine if the design is ready for production and if the producer has accomplished adequate production planning" (DAG).

The primary difference between PRR and TRR is that the system test results are available prior to PRR. If changes are made to the system in response to test results, it will be necessary to revisit TRR tasks.

Any evidence provided by system test results should be incorporated into the assurance case prior to PRR. Verify that the system test results when combined with previously obtained evidence support the arguments in the assurance case and that those arguments justify the claims made by the assurance case.

To successfully complete the SVR and/or PRR reviews, the following activities provide evidence:

- Incorporation of the results of system test for weaknesses and their associated vulnerabilities into the assurance case. Verification that the system weaknesses and vulnerabilities have been baselined and appropriately document (See IA control: VIVM-1).
- Verification that the developed system uses comprehensive test procedures to test any and all patches and upgrades required throughout its life cycle (See IA control: DCCT-1).
- Incorporation of the results of any testing, using industry tools and test cases, for any binary or machine-executable public domain software products with no warranty and no source code being used in the system (See IA control: DCPD-1).
- Evaluation of the relevant test results to obtain the evidence required to build the assurance case from the following list of defensive functions of a system as well as assurance mechanisms that address security, partitioning, access, and traceability mechanisms:
  - Evaluation of the protection mechanisms with each external interface and associated security requirements using test results as well as other evidence (See IA control: DCFA-1).
  - Evaluation of the adequacy of security best practices of such functions as identification/authentication (individual and group) using DoD PKI, logon including single sign-on, PKE, key management, smart card, and biometrics (see IA controls: as per DoDI 8500.2 DCBP-1, IATS-2, IAAC-1, IAGA-1, IAIA-2, IAKM-3, ECLO-2, ECPA-1).
  - Evaluation of user interface services are logically or physically separated from data storage and management services. This is particularly important in high assurance systems (see IA controls: DCPA-1).
  - Evaluation that security support structure is isolated by means of partitions, domains, etc., including control of access. Examples include sandboxes, jails, and separation kernels (see IA controls: DCSP-1). Evaluate the system's ability to achieve confidentiality, integrity, availability, and non-repudiation (See IA controls: DCNR-1 (non-repudiation), ECCR-2 (confidentiality for data at rest), ECCT-2 (confidentiality for data in transit), ECWN-1 (confidentiality for wireless implementations in accordance with DOD policy)).
  - Evaluation that stored, processed, transmitted, or displayed data are appropriately marked, labeled, and encrypted (see IA control ECML-1, ECNK-1, ECNK-2).
- Evaluation that there is adequate implementation of the following access control mechanisms:
  - Least privilege (see IA control: ECLP-1).

- Data, including classified data, are accessed, changed, and logged only by authorized personnel with appropriate 'need-to-know'(See IA control: ECCD-2, ECAN-1).
- Remote access for privileged functions permitted for completing operational needs is strictly controlled, and remote access to user functions is mediated through a managed access control point (see IA controls; EBRP-1, EBRU-1).
- All authorizations to information contained within an object are revoked prior to initial assignment, allocation, or reallocation (see IA control ECRC-1).
- Examples include limiting privileges to changing production code and data as well as preventing unauthorized access.

- Check system for compliance with ports, protocols, and services guidance according to DoD information systems requirements (see IA control: DCPP-1).
- Evaluation of the implementation of automated continuous on-line audit trail mechanism to provide the following capabilities (see IA controls: ECAT-2, ECRG-1, ECTP-1, ECLC-1, ECAR-3):
  - Protection of audit trail from unauthorized access, modification, or deletion
  - Tools for monitoring, analysis, review, and report generation of audit records
  - Automatic recording, creation, deletion, or modification of security labels (where appropriate)
- Verification that System initialization, shutdown, aborts, and recovery are configured to ensure the system remains in a secure and verifiable state (see IA control: DCSS-2, COTR-1)
- Verification that all critical COTS/GOTS elements have been evaluated or validated through an approved source, such as NIAP evaluation and validations programs or NSA evaluation (see IA control: DCAS-1)
  - Check that configuration management supports traceability and protection of each configuration item (See IA Controls: DCPR-1, DCSL-1, ECPC-2, ECSD-2, ECVP-1, PESL). Check that the integrity mechanism required for all critical COTS/GOTS elements functions properly to assure the integrity of all transmitted information and to detect and prevent the hijacking of communications sessions (see IA control: ECTM-2).
  - Where required, check that COTS/GOTS with high-robustness characteristics are functioning properly, e.g., when information transits networks to lower classification levels (see IA control: DCSR-3).
- Ensure that the maintenance plan includes a patch management process, which must include:
  - Method(s) to test patches adequately before deploying them, including testing to ensure that assurance properties are maintained
  - Method(s) for rapid deployment of "emergency" patches without full testing
  - An adjudication approach for selecting the appropriate method that considers system-assurance requirements and risks
  - Patch authentication process to be used by recipients (to ensure that rogue patches are not applied)
- Ensure that an incident response plan exists that identifies the responsible Computer Network Defense (CND) Service Provider in accordance with DoD Instruction O-8530.2, defines reportable incidents, outlines a standard operating procedure for incident response to include Information Operations Condition (INFOCON), provides for user training, and

establishes an incident response team. The plan is exercised at least every 6 months (per DODI 8500.2 VIIR-2, Incident Response Planning).

- Ensure that a DoD reference document, such as a security technical implementation guide or security recommendation guide, is used. This guide constitutes the primary source for security configuration or implementation guidance for the deployment of newly acquired IA- and IA-enabled IT products that require use of the product's IA capabilities. If a DoD reference document is not available, the system owner works with Defense Information Systems Agency (DISA) STIGS (http://iase.disa.mil/stigs/stig/index.html) or the National Security Agency (NSA) guidelines (http://www.nsa.gov/snac) to draft configuration guidance for inclusion in a Departmental reference guide. The security recommendation guide should instruct the system to depend on the elements that are most assured, and limit dependence on those less assured (per DODI 8500.2 DCCS-2, Configuration Specifications).

For more detailed guidance, see the Transition and Validation Process sections (3.4.7 and 3.4.8) of this document.

4.3.3.8 Milestone C

This is the decision point for exiting system development and demonstration. The exit criteria of the milestone decision authority (MDA) should include system assurance criteria. At this point, as part of reviewing the exit criteria, ensure that system assurance has been considered and addressed throughout its development. Examine the system assurance case and appropriate supporting evidence, to ensure that the system has demonstrated adequate assurance.

For more detailed guidance, see the Transition, and Validation process sections (3.4.7 and 3.4.8) of this document.

4.3.4 Production, Deployment, Operations, and Support Phases

System assurance must be maintained in production, deployment, operations, and support. Some artifacts should include assurance considerations and are updated throughout system production, deployment, operations, and support. These include the Systems Engineering Plan (SEP), Test and Evaluation Master Plan (TEMP), the System Threat Assessment, Test Reports, and the Risk Assessment. When significant new threats or vulnerabilities are identified, an abbreviated version of the system life cycle is necessary to address them; a process must be in place to manage such reports without unintentionally disclosing this information. This process should update other artifacts (e.g., architecture/design and assurance case) so the necessary traceability for assurance is maintained.

"The program manager may conduct another TRR prior to Initial Operational Test and Evaluation. The OTRR is a multi-disciplined product and process assessment to ensure that the "production configuration" system can proceed into Initial Operational Test and Evaluation with a high probability of successfully completing the operational testing. Successful performance during operational test generally indicates that the system is suitable and effective for service introduction" (DAG).

To successfully complete the OTR, all of the criteria for the TRR (See section 4.3.3.6) must be completed with the following additions or deletions.

- Source code static analysis is typically not performed again for OTRR, but binary analysis is performed, if appropriate.
- Ensure that the documented operational procedures for addressing attack scenarios are inspected, any rework performed, and approved for use in operational test.
- Development of the expected results that will be satisfactory evidence to complete the assurance case and meet the system assurance requirements.
- Verification that (1) assurance-specific static (binary) analysis and (2) peer reviews addressing assurance criteria have been completed. Verify that another party (such as a peer) performed static (binary) analysis and peer review. Ensure that detected defects have been corrected or mitigated, and appropriately documented.
- Development of (test input) data to be used in tests of system assurance requirements, based on the specific instances of system assurance scenarios.
- Establishment of the expected results that will be satisfactory evidence to complete the assurance case and meet the system assurance requirements.
- Use of system implementation information to identify, test, verify, and log weaknesses and their associated vulnerabilities. Continue refining the baselined attack scenarios of the identified threats. Ensure that all appropriate system patches are up to date (see IA control: VIVM-1).
- Ensure the developed system has a set of comprehensive test procedures to test that any and all patches and upgrades do not adversely affect system security and assurance (see IA control: DCCT-1).
    - Development and maintenance of a listing of hosting enclaves. This includes the collection of evidence deployment planning and coordination as well as the exchange of connection rules and requirements (see IA control: DCID-1).
    - Reevaluation of integrated mobile code for acceptable risk, with results used to verify implementation to support assurance needs (see IA controls: DCMC-1).
    - Verification, using industry tools, that no binary or machine-executable public domain software products with no warranty and no source code are being used in the system (see IA control: DCPD-1).
- Analysis of the system for weakness and vulnerabilities using static (binary) analysis tools to identify such flaws as buffer overruns and cross-site scripting issues, etc. (See IA controls; DCSQ-1).
- Ensure that the system incorporates the following security, partitioning, access, and traceability mechanisms that are key defensive functions to assure the system:
    - Reverification that protection mechanisms with each external interface and associated security requirements are implemented to meet system requirements (See IA control: DCFA-1).
    - Reverification that the system incorporates best security practices such as identification/authentication (individual and group) using DoD PKI, logon including single sign-on, PKE, key management, smart card, and biometrics (see IA controls: as per DoDI 8500.2 DCBP-1, IATS-2, IAAC-1, IAGA-1, IAIA-2, IAKM-3, ECLO-2, ECPA-1, Best Security Practices).
    - Reverification that user interface services are logically or physically separated from data storage and management services to meet system requirements (see IA controls: DCPA-1).

- Reverification that security support structure is isolated by means of partitions, domains, etc., including control of access (see IA controls: DCSP-1).
- Reverification, as required, that the system meets confidentiality, integrity, availability, and non-repudiation requirements (see IA controls: DCNR-1 (non-repudiation), ECCR-2 (confidentiality for data at rest), ECCT-2 (confidentiality for data in transit), ECWN-1 (confidentiality for wireless implementations in accordance with DOD policy)).
- Information and DoD information systems that store, process, transmit, or display data are appropriately marked, labeled, and encrypted (see IA control ECML-1, ECNK-1, ECNK-2).
- Access control mechanisms such as the following must be reverified to meet system requirements:
- Least privilege (see IA control: ECLP-1).
- Ensure data, including classified data, is accessed, changed. and logged only by authorized personnel with appropriate 'need-to-know' (see IA control: ECCD-2, ECAN-1).
- Remote access for privileged functions is permitted for completing operational needs and is strictly controlled, and remote access to user functions is mediated through a managed access control point (see IA controls; EBRP-1, EBRU-1).
- All authorizations to information contained within an object are revoked prior to initial assignment, allocation, or reallocation (see IA control ECRC-1).

- Reverification that automated continuous on-line audit trail mechanisms provide the following (see IA controls: ECAT-2, ECRG-1, ECTP-1, ECLC-1, ECAR-3):
  - Protection of audit trail from unauthorized access, modification, or deletion
  - Tools for monitoring, analysis, review, and report generation of audit records
  - Automatic recording, creation, deletion, or modification of security labels (where appropriate)
- Configuration of system initialization, shutdown, aborts, and recovery to insure the system remains in a secure and verifiable state (see IA control: DCSS-2, COTR-1).
- Reverification that the integrity mechanisms required for all critical COTS/GOTS elements function properly to assure the integrity of all transmitted information and to detect and prevent the hijacking of communications sessions (see IA control: ECTM-2).
- Where required, reverification that COTS/GOTS with high robustness characteristics are functioning properly, e.g., when information transits networks to lower classification levels. (see IA control: DCSR-3).
- Reverification that configuration management issues (that address assurance) support traceability and protection of each configuration item noted (see IA Controls: DCPR-1, DCSL-1, ECPC-2, ECSD-2, ECVP-1, PESL-1).
- Reverification that the system is compliant with ports, protocols, and services guidance according to DoD information systems requirements. This is one of the last opportunities before the system is deployed to lock it down or to address security issues with interfaces (see IA control: DCPP-1).

For more detailed guidance, see the Verification, Transition, and Validation Process sections (3.4.6, 3.4.7, and 3.4.8) of this document.

4.3.4.1 Initial deployment and operation

Although there is no review process within the DAG for initial deployment and operation, it is important to note that there are significant assurance activities during deployment and operation. For more detailed guidance, see the Sections 3.4.7, Transition, and 3.4.9, Operation and Training.

4.3.4.2 Physical Configuration Audit

"The PCA [Physical Configuration Audit] is conducted around the time of the full rate production decision. The PCA examines the actual configuration of an item being produced. It verifies that the related design documentation matches the item as specified in the contract. In addition to the standard practice of assuring product verification, the PCA confirms that the manufacturing processes, quality control system, measurement and test equipment, and training are adequately planned, tracked, and controlled. The PCA validates many of the supporting processes used by the contractor in the production of the item and verifies other elements of the item that have been impacted/redesigned after completion of the System Verification Review (SVR)" (DAG).

To successfully complete the PCA, ensure the following system assurance items were satisfactorily completed:

- Ensure that counterfeit/substandard hardware/software elements are not incorporated into the final system. Counterfeit hardware has, in some circumstances, become a serious problem.
- Ensure that any assurance-specific audits have been completed and documented.
- Ensure that any source code delivered is actually the source code used by the system (where it is claimed to be).
- Ensure that packaging includes relevant seals to verify authenticity and impede tampering.

For more detailed guidance, see the Verification Process section (3.4.6) of this document.

4.3.4.3 Full-Rate Production Decision Review

"Based on the results of testing initial production articles and refining manufacturing processes and support activities, knowledge prior to committing to full-rate production should indicate the product is operationally capable; lethal and survivable; reliable; supportable; and producible within cost, schedule, and quality targets" (DAG 11.5).

In Full-Rate Production Decision Review (FRPDR):

- Ensure that any critical element assurance methods can scale up to the full rate.
- If additional production lines or outsourced manufacturers are added, ensure that they meet the same assurance criteria.

For more detailed guidance, see the Transition and Validation process sections (3.4.7 and 3.4.8) of this document.

4.3.4.4 Post-Deployment Review and In-Service Review

"Post-Deployment Review (PDR) is used by the program manager beginning at Initial Operational Capability (IOC) to verify whether the fielded system continues to meet or exceed thresholds and objectives for cost, performance, and support parameters approved at full-rate production" (DAG).

In-Service Review (ISR) "is a multi-disciplined product and process assessment to ensure that the system under review is operationally employed with well-understood and managed risk. This review is intended to characterize the in-service technical and operational health of the deployed system. It provides an assessment of risk, readiness, technical status, and trends in a measurable form" (DAG).

To successfully complete these reviews, ensure the following system assurance items were satisfactorily completed:

- Examination of changes in the system configuration and environment; evaluation if the assurance claims continue to be valid under these changes. If not, take corrective action. An example of such changes is disabling infrastructure defensive functions, such as firewalls, so the system can work in its environment.
- Review of system operational performance with respect to its ability to withstand attack, particularly by examining logs of attacks and performance during attacks. Do this annually in support of 8500.2 IA control DCAR-1, which focuses on procedural review, with the goal of supporting uninterrupted operations. Compare system with best practices, as discussed in 8500.2 IA control DCBP-1.
- Determination whether new threats have developed, compared to already-identified threats (such as in the System Threat Assessment Report (STAR)). Information sources include:
  - review of attempted attacks on this system
  - review of attacks on other similar systems
  - use of all-source information on attacker motivations and tactics
- Reviewed patch management process, and its evidence, to determine that it is adequate and being followed; see the PCA section for specific requirements.
  - For example, for COTS/GOTS products included in the system, determine what patches have not been applied, and of those, which ones present potential vulnerabilities.
- Ensure that the incident response plan noted in section "System Verification Review (SVR) and Production Readiness Review (PRR)" is updated as required and is exercised at least every 6 months (per DODI 8500.2 VIIR-2, Incident Response Planning).

For more detailed guidance, see the Transition and Validation process sections (3.4.7 and 3.4.8) of this document.

4.3.4.5 Sustainment

It is important to note that there are significant assurance activities during sustainment. For more detailed guidance, see section 3.4.10, Maintenance.

4.3.4.6 Disposal

Ensure that clearing/sanitizing/destruction follow DoD requirements defined in DoD 5200.1-R. For general system assurance processes related to this, see the Disposal process section 3.4.11 (per DoDI 8500.2 PECS-2, Clearing and Sanitizing, and PEDD-1, Destruction).

4.4 Supporting Processes

This section presents controls and guidance that are directly applicable to system assurance, and are repeated across multiple life cycle processes. Some deliver artifacts to various DAG reviews (e.g., PDR, CDR, PRR, and ISR) that are valuable as evidence for assurance claims.

At multiple review points:

- Ensure that all systems are under the control of a chartered Configuration Control Board (CCB) that meets regularly according to DCPR-1. The IAM is a member of the CCB (per DoDI 8500.2 DCCB-2, Control Board).
- Configure software according to appropriate security guidelines, such as the Defense Information Systems Agency (DISA) STIGS (http://iase.disa.mil/stigs/stig/index.html) and the NSA security configuration guides (http://www.nsa.gov/snac). These guides provide configuration settings to harden software against attack and should be used for software employed in both development and operation.
- Ensure that application programmer privileges to change production code and data are limited and are reviewed every 3 months (per DODI 8500.2 ECPC-2, Production Code Change Controls).
- Ensure that the development environment and system being developed do not use services they cannot trust. This is particularly important to review during PDR, CDR, TRR, and ISR. Do not connect to systems you don't trust. Examples are cited in DoDI 8500.2:
  - *ECIM-1 Instant Messaging.* Instant messaging (IM) traffic to and from IM clients that are independently configured by end users and that interact with a public service provider is prohibited within DoD information systems. Both inbound and outbound public service IM traffic is blocked at the enclave boundary. Note: This does not include IM services that are configured by a DoD AIS application or enclave to perform an authorized and official function.
  - *ECVI-1 Voice over Internet Protocol.* Voice over Internet Protocol (VoIP) traffic to and from workstation IP telephony clients that are independently configured by end users for personal use is prohibited within DoD information systems. Both inbound and outbound individually configured VoIP are blocked at the enclave boundary. Note: This does not include VoIP services that are configured by a DoD AIS application or enclave to perform an authorized and official function.
- Ensure that a comprehensive vulnerability management process (VIVM) that includes the systematic identification and mitigation of software and hardware vulnerabilities is in place. VIVM is applicable to the following DAG Reviews: SFR, PDR, CDR, TRR, PRR, ISR. Wherever system capabilities permit, mitigation is independently validated through inspection and automated vulnerability assessment or state management tools. To keep systems up to date, use the JTF-GNO IAVA (http://iase.disa.mil/index2.html) database and such resources as US CERT Technical Bulletins and Advisories (http://www.us-cert.gov/). Such tactical efforts as vulnerability assessment tools are acquired, personnel are appropriately trained, procedures are developed, and regular internal and external

assessments are conducted. For improved interoperability, preference is given to tools that express vulnerabilities in the Common Vulnerabilities and Exposures (CVE) naming convention and use the Open Vulnerability Assessment Language (OVAL) to test for the presence of vulnerabilities (per DODI 8500.2 VIVM-1, Vulnerability Management). Ensure that mobile code, if used, conforms to the DoD mobile code policy noted in DODI 8500.2 DCMC-1, Mobile Code, and the memorandum "Guidance for the Use of Mobile Code Technologies in Department of Defense (DoD) Information Systems." Mobile code can add significant risks, since it is often very difficult to assure. Therefore, its use needs to be limited, or it must be executed within partitioned environments that limit access (e.g., sandboxes, virtual machines, jails, containers). This should be specifically checked during PDR and CDR (per DODI 8500.2 DCMC-1 Mobile Code).

- Ensure that access to both the development environment and the operational system is gained only through the presentation of an individual identifier (e.g., a unique token or user logon ID and password). This helps manage risk against attack of both the developmental environment and operational system. This is per DoDI 8500.2 IAIA-2, Individual Identification and Authentication. For the development environment, IAIA-2 must be implemented before the environment is used. For the operational system, the method for meeting IAIA-2 must be defined by CDR, and implemented by TRR. To meet these criteria:
  - For systems utilizing a logon ID as the individual identifier, passwords are, at a minimum, a case-sensitive, 8-character mix of upper-case letters, lower-case letters, numbers, and special characters, including at least one of each (e.g., emPagd2!). At least four characters must be changed when a new password is created. Deployed/tactical systems with limited data input capabilities should implement these measures to the extent possible.
  - Registration to receive a user ID and password includes authorization by a supervisor, and is done in person before a designated registration authority. Multiple forms of certification of individual identification such as documentary evidence or a combination of documents and biometrics are presented to the registration authority.
  - Additionally, to the extent capabilities permit, system mechanisms are implemented to enforce automatic expiration of passwords and to prevent password reuse, and processes are in place to validate that passwords are sufficiently strong to resist cracking and other attacks intended to discover a user's password.
  - All factory-set, default, or standard-user IDs and passwords are removed or changed.
  - Authenticators are protected commensurate with the classification or sensitivity of the information accessed; they are not shared; and they are not embedded in access scripts or stored on function keys. Passwords are encrypted both for storage and for transmission (this list per DODI 8500.2 IAIA-2, Individual Identification and Authentication).
- Ensure that access to all DoD information is determined by both its classification and user need-to-know. This increases assurance by reducing the opportunities for exploitation of weaknesses in the development environment and operational system (per DODI 8500.2 ECAN-1, Access for Need-to-Know):
  - Need-to-know is established by the Information Owner and enforced by discretionary or role-based access controls. Access controls are established and enforced for all shared or networked file systems and internal Web sites, whether classified, sensitive, or unclassified.

- All internal classified, sensitive, and unclassified Web sites are organized to provide at least three distinct levels of access:
  - ♦ Open access to general information that is made available to all DoD authorized users with network access. Access does not require an audit transaction.
  - ♦ Controlled access to information that is made available to all DoD authorized users upon the presentation of an individual authenticator. Access is recorded in an audit transaction.
  - ♦ Restricted access to need-to-know information that is made available only to an authorized community of interest. Authorized users must present an individual authenticator and have either a demonstrated or validated need-to-know. All access to need-to-know information and all failed access attempts are recorded in audit transactions (per DODI 8500.2 ECAN-1, Access for Need-to-Know).
- These access restrictions, including audits to verify them, are used as evidence in the assurance case.

- Ensure that an automated, continuous on-line monitoring and audit trail creation capability is established for both the development environment and operational system. This capability should be deployed to immediately alert personnel of any unusual or inappropriate activity with potential IA implications, and with a user-configurable capability to automatically disable the system if serious IA violations are detected. For the development environment, this must be implemented before the environment is used. For the operational system, the method for meeting this must be defined by CDR and examined during PRR, TRR, and ISR (per DODI 8500.2 ECAT-2, Audit Trail, Monitoring, Analysis and Reporting).
- Ensure that clearing/sanitizing/destruction follow DoD requirements defined in DoD 5200.1-R. For general system assurance processes related to this, see the Disposal process section 3.4.11 (per DoDI 8500.2 PECS-2, Clearing and Sanitizing, and PEDD-1, Destruction).
- Ensure that procedures are implemented for the proper handling and storage of information, such as end-of-day security checks, unannounced security checks, and, where appropriate, the imposition of a two-person rule within the computing facility. For assurance, some critical elements require 2-person controls (where an element cannot be created or modified without a second person reviewing and verifying it) (per DoDI 8500.2 PESP-1, Workplace Security Procedures). To do this:
  - Ensure that documents and equipment are stored in approved containers or facilities with maintenance and accountability procedures that comply with DoD 5200.1-R (DoDI 8500.2 PESS-1, Storage).

4.4.1 Periodic Reports

Periodic reports should include information on new threats identified or new actions taken to counter threats. If there is a significant risk that system assurance requirements will not be met, this must be highlighted in the periodic reports.

The operating procedures that were approved at product readiness review (PRR) must include information assurance verification test cases that are run daily, weekly, and monthly to ensure that the system is functioning as intended and continues to meet the DoD compliance requirements. Any deviations should be recorded in a log, corrective actions taken, and a more robust set of tests executed to revalidate the system.

4.4.2 Anti-Tamper

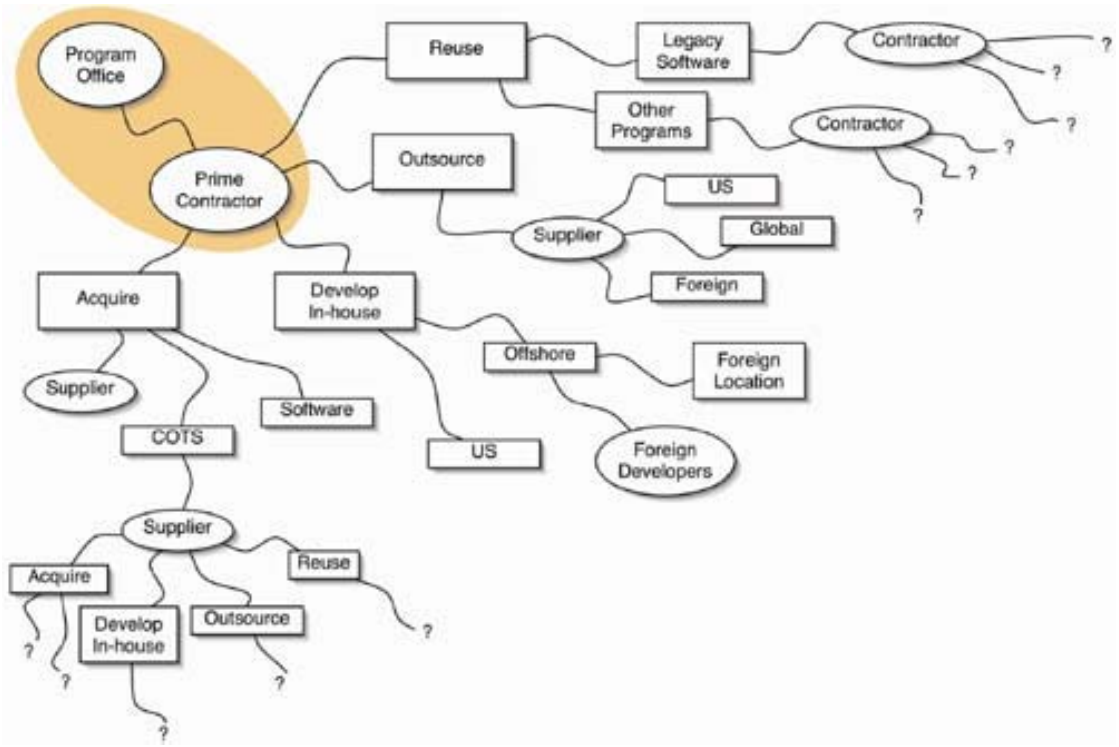
The purpose of anti-tamper mechanisms is "to deter the reverse engineering and exploitation of our military's critical technology, in order to impede technology transfer, stop alteration of system capability, and prevent the development of countermeasures to U.S. systems," according to "Anti-Tamper Program" published by the DoD Anti-tamper Executive Agent.

Anti-tamper as a countermeasure is addressed in Section 3. Anti-tamper techniques should be implemented into the design to protect Critical Program Information (CPI) and Critical Technology (CT) per the Anti-Tamper Plan that is approved by the program manager. Specific anti-tamper program guidance can be obtained by contacting the Office of Primary Responsibility (OPR) for the Service or Agency. DoD suppliers can obtain DoD-specific anti-tamper information from the following:

- DoD Anti-tamper Executive Agent Web site: http://at.dod.mil
- The-OPR for anti-tamper for a respective Service or agency.
- Defense Acquisition University (DAU) course on anti-tamper (unclassified). DAU Continuous Learning Website: http:///clc.dau.mil Course CLE 022.
- AFRL RYT Anti-Tamper and Software Protection Fact Sheet: http://www.wpafb.af.mil/library/factsheets/factsheet_print.asp?fsID=6308.

4.4.3 Supplier Assurance

Innovation and technology are global. The DoD must be able to employ technologies developed worldwide, without exposing itself to excessive risk from subversion of the supply chain throughout the system life cycle. Figure 4-2 illustrates some of the combinations that can complicate the supply chain. Program offices may award contracts to prime contractors, who typically use a combination of custom work (including in-house development, outsourcing, and subcontracting of custom work) along with use of OTS elements (including COTS and GOTS). The key issue is that knowing the identity of the prime contractor, or the first layer of subcontractors, does not adequately describe the true supply chain.



Source: Walker (2005)

Figure 4-2 Supply Chain

There is a risk that lower-tier suppliers (down to the level of individual people) may insert, intentionally or intentionally, vulnerabilities that impact the missions that the systems are to support. Thus, there is a need for management of this risk (including mitigation). Unfortunately, typical contracting approaches provide little insight into the supply chain, leading to great difficulty in decision making and risk management.

Therefore, there is a need to gain greater knowledge about the supply chain, to enable better decision making and risk management. Acquirers need more relevant and timely knowledge about their supply chain. Some suppliers are riskier than others. This does not mean that they cannot be used, but it does mean that additional countermeasures may be needed (e.g., greater transparency of the supplied element and/or its processes, limited privileges, blind buys, etc.).

At the apex of the supply chain is a system integrator, who selects OTS elements to be included in the component, develops/configures custom elements, or subcontracts to lower-tier system integrators who do the same. Thus, there are two major types of elements:

- An OTS element, including COTS and GOTS. This often has the benefit of being used over time in many different applications and markets, providing a wealth of information about its risks through its vetting by the marketplace. It may have been vetted through other mechanisms as well, such as certifications, evaluations, etc. While vulnerabilities may be embedded in the OTS element, varying use may expose them, and since the OTS supplier often does not know exactly how or where the element is used, targeted attacks are often more difficult to implement. OTS elements are often far less costly than custom development. Depending on the needs of the system, however, existing OTS options might not provide the properties needed for the mission.

- A custom element, developed by a system integrator (at the top or lower tier). Such elements are typically not used in many different applications or markets. The developers of these elements typically have great insight into exactly how the element will be used, including its operational environment and its associated requirements. Vulnerabilities in custom elements present greater risks, because there is usually no marketplace vetting, and a malicious supplier could determine precisely how to cause the most damage to the system.

In some cases, an element may start as a custom element and then transition to become an OTS element. In other cases, an OTS element may require modification for use, and that modified element becomes custom.

Any supplier could introduce a vulnerability into a system. However, the suppliers of custom elements should be examined especially carefully, because there is no vetting from the marketplace and it easier for them to insert targeted attacks. Thankfully, it is also possible to impose additional requirements (e.g., on people, process, product/services, and tools) on custom elements to at least partly address this risk.

The following subsections discuss element criticality, riskiness of supplier, handling OTS elements, handling custom elements, trusted integrated circuits (ICs), example countermeasures, and where to find additional information.

4.4.3.1 Element criticality

Determine the criticality of the element to the overall system and its missions; see table 3-4 on system criticality for an example of how to determine criticality, though for DoD systems and organizations the properties and tolerances for criticality may be different.

The criticality of the system or element should be reflected in its required robustness. For software, DoD Instruction 8500.2 identifies three levels of robustness: high, medium, and basic, which may be leveraged to support assurance claims. DoDI 8500.2 DCSR-3, DCSR-2, and DCSR-1 (Specified Robustness – High, Medium, and Low) state the following:

- DCSR-3 Specified Robustness – High. Only high-robustness GOTS or COTS IA and IA-enabled IT products are used to protect classified information when the information transits networks that are at a lower classification level than the information being transported. High-robustness products have been evaluated by NSA or in accordance with NSA-approved processes. COTS IA and IA-enabled IT products used for access control, data separation, or privacy on classified systems already protected by approved high-robustness products at a minimum, satisfy the requirements for basic robustness. If these COTS IA and IA-enabled IT products are used to protect National Security information by cryptographic means, NSA-approved key management may be required.
- DCSR-2 Specified Robustness – Medium. At a minimum, medium-robustness COTS IA and IA-enabled products are used to protect sensitive information when the information transits public networks or the system handling the information is accessible by individuals who are not authorized to access the information on the system. The medium-robustness requirements for products are defined in the Protection Profile Consistency Guidance for Medium Robustness published under the IATF. COTS IA and IA-enabled IT products used for access control, data separation, or privacy on sensitive systems already protected by approved medium-robustness products, at a minimum, satisfy the requirements for basic robustness. If these COTS IA and IA-enabled IT products are used

to protect National Security information by cryptographic means, NSA-approved key management may be required.

- DCSR-1 Specified Robustness – Basic. At a minimum, basic-robustness COTS IA and IA-enabled products are used to protect publicly released information from malicious tampering or destruction and ensure its availability. The basic-robustness requirements for products are defined in the Protection Profile Consistency Guidance for Basic Robustness published under the IATF.

DoD Instruction 8500.2, DCDS-1 (Dedicated IA Services), requires that "Acquisition or outsourcing of dedicated IA services such as incident monitoring, analysis and response; operation of IA devices such as firewalls; or key management services are supported by a formal risk analysis and approved by the DoD Component CIO" (per DoDI 8500.2, DCDS-1).

When an acquisition program containing CPI is initiated, the program manager should request a multidisciplinary counterintelligence (MDCI) threat assessment. The MDCI threat focuses on how the opposition sees the program, and on how to counter the opposition's collection efforts. The MDCI threat assessment will provide the program manager with an evaluation of foreign collection threats to specific program or project technologies, the impact if that technology is compromised, and the identification of related foreign technologies that could impact program or project success. When gathering information to meet the needs described in this chapter, intelligence and CI organizations must comply with DoD Directive 5240.1 and DoD 5240.1-R. Information gathered by non-intelligence community entities must comply with DoD Directive 5200.27. Traditionally CPI and MDCI have focused on confidentiality; for system assurance purposes, consider extending MDCI for assuring integrity, availability, authentication, accountability (including non-repudiation), and auditability (e.g., to ensure that elements used in systems are not subverted and/or externally controlled). For more information, see section 4.2, Protection Plan Implementation.

For more information on draft procedures for research and technology protection, see DoD 5200.39-R, published March 2006.

4.4.3.2 Riskiness of supplier

Consider the security-related practices and procedures of suppliers when selecting them, both for custom and OTS elements. In particular, judge to determine the risk of the suppliers' processes allowing intentional or unintentional vulnerabilities that impact missions. Suppliers of highly critical and/or custom elements may require more rigorous and broader evaluation.

Sources of information that can assist in evaluating the suppliers' security-related practices and procedures include:

- Government-Industry Data Exchange Program (GIDEP), a cooperative activity between government and industry participants seeking to reduce or eliminate expenditures of resources by sharing technical information essential during research, design, development, production, and operational phases of the life cycle of systems, facilities, and equipment.[3]

[3] More information on the DoD-sponsored program can be found at http://www.gidep.org/

- Qualified Product List (QPL), Qualified Manufacturers List (QML), or Qualified Product Database (QPD). The Defense Supply Center Columbus (an arm of the Defense Logistics Agency) has focused on maintaining a good base of suppliers that have successfully demonstrated their products meet the specified performance, quality, reliability, maintainability, and testability (system health) levels via the DoD product Qualification program.[4]
- Operational Test & Evaluation (OT&E) reports—both on the product being considered, if available, and on recent, similar projects by the same supplier. Particularly relevant are any information assurance assessments on the products' abilities to protect, detect, react/respond, and restore.

Certain types of suppliers present additional risks:

- Software is high risk when there is no known owner/supplier who can provide updates, the software is only distributed as binary or machine code, and the source code cannot be read, modified, and redistributed. Such products should be assessed, as required, for information assurance impacts, and may be approved for use only by the Designated Approval Authority (DAA) (per DoDI 8500.2 DCPD-1, Public Domain Software Controls).
- Outsourced IT-based processes have special assurance concerns (see DoDI 8500.2 E2.1.17.3). These are outsourced business processes supported by private sector information systems, outsourced information technologies, or outsourced information services. Outsourced IT-based processes' assurance depends on the supplier's infrastructure, but the customer will not control this infrastructure, and often cannot gain adequate visibility of it. Therefore, the assurance of outsourced elements must consider how adequate assurance can be gained given this visibility.

4.4.3.3 Off-the-shelf

In the system life cycle, analysis of alternatives is used to determine whether to use OTS or custom elements (and which ones). Note that OTS elements are not typically modified (other than some supplier-provided configuration capabilities). Typically there is relatively little visibility into a supplier's implementation and processes.

See section 3.1.1.1, "Assurance Considerations in Acquisition of COTS & GOTS," for more information on handling supplier assurance for OTS.

4.4.3.4 Custom

System assurance for custom elements differs from OTS. Most system integrators (at the top or lower tier) who develop these custom elements have visibility into the mission and how the system is used to support it , the associated countermeasures, and the environment in which the system resides. Although this knowledge is valuable for the development of a custom system, it could be used by attackers to facilitate targeted attacks. The customer and the system integrator must collaborate to increase the assurance through the supply chain (see section 3.1.1.2 for more detail on the supply chain).

---

[4] http://www.dscc.dla.mil/offices/sourcing_and_qualification/default.asp

The aspects of supplier assurance that must be addressed by both customers and system integrators include preparation and communication of requirements (Request for Proposal), supplier selection, people, products, and tools.

Prepare and Communicate Requirements (Request for Proposal)

Note special assurance-related requirements directly in the RFP, e.g., for requiring cleared personnel for development. Demand that suppliers provide their elements "secure-by-default" (DSB 2007).

Select System Integrator

Consider the security-related practices and procedures of custom suppliers when selecting a systems integrator. In particular, judge to determine the risk of the supplier (or the supplier's suppliers) causing intentional or unintentional vulnerabilities that impact missions. Suppliers of custom elements often require more rigorous and broader evaluation, particularly if they are supplying highly critical elements.

People

Consider if critical custom elements (hardware and software) developed for DoD systems should be developed only by cleared personnel. Note that critical elements can impact the system's missions. See the recommendations of *Mission Impact of Foreign Influence on DoD Software* (DSB 2007).

Individuals requiring access to classified information must be processed for access authorization in accordance with DoD personnel security policies (per DoDI 8500.2 PRAS-2, Access to Information).

Ensure that maintenance will be performed only by authorized personnel. Except as authorized by the DAA, personnel who perform maintenance on classified DoD information systems must be cleared to the highest level of information on the system. Cleared personnel who perform maintenance on classified DoD information systems require an escort unless they have authorized access to the computing facility and the DoD information system (see DoDI 8500.2 PRMP-2, Maintenance Personnel).

Process

DoD acquisition standards must be followed. Per DoDI 8500.2, "The acquisition of all IA- and IA-enabled GOTS IT products is limited to products that have been evaluated by the NSA or in accordance with NSA-approved processes. The acquisition of all IA- and IA-enabled COTS IT products is limited to products that have been evaluated or validated through one of the following sources - the International Common Criteria (CC) for Information Security Technology Evaluation Mutual Recognition Arrangement, the NIAP Evaluation and Validation Program, or the FIPS validation program. Robustness requirements, the mission, and customer needs will enable an experienced information systems security engineer to recommend a Protection Profile, a particular evaluated product, or a security target with the appropriate assurance requirements for a product to be submitted for evaluation" (per DoDI 8500.2 DCAS-1, Acquisition Standards). These other evaluations become evidence to support system assurance claims. (See also DCSR-1.)

Product

As noted earlier, products are to be evaluated through a variety of tools and methods (including static analysis of binary and/or source code, dynamic test tools such as fuzzers, etc.). As noted earlier, software that is distributed only as binary or machine code where there is no known owner/supplier who can provide updates, and the source code cannot be read, modified, and redistributed, should typically be avoided. If such software is used, it should undergo a more rigorous evaluation before approval, and/or have a plan to migrate away from it (see DoDI 8500.2 DCPD-1, Public Domain Software Controls.)

Tools

See “Tools” in Section 3.1.1.

4.4.3.5 Trusted integrated circuits

To a large extent, integrated circuit (IC) assembly and fabrication have moved overseas, making state-of-the-art IC technology available internationally. ICs are complex, making it difficult to determine if the approved design has been altered or subverted. Each phase of the IC life cycle, including design, fabrication, testing, packaging, distribution, and maintenance, is vulnerable to attacks by adversaries. Thus, ensuring trust in defense ICs is a matter of managing the risk of being attacked, and finding ways to protect the IC supply chain at reasonable cost by putting a system of trust in place.

One approach is to choose a “trusted supplier” appropriate for the sensitivity of the ICs, whether custom or OTS. The basic idea behind selecting a trusted supplier is that the supplier can be trusted to (1) produce quality IC chips for defense and national security applications, (2) protect the design and intellectual property contained in the fabricated chips (for custom ICs), and (3) prevent tampering. For those reasons, the potential supplier must have a system for quality management and quality control in place. Quality means the fabricated chips would perform as expected in the design, and that includes reliability. Quality cannot be inserted at the last stage of manufacturing but must be ensured at every stage of fabrication. A part of the trust is to ensure that the chips have not been intentionally or unintentionally modified or tampered with, and that no unintended devices and circuits have been inserted into the chips. Thus, the trusted supplier must have in place a system that tracks the chips throughout the manufacturing and distribution process. A record must be kept so that the status of the chips and data can be tracked with an electronic or paper trail. The industrial best practices are often in place in the best companies. These attributes apply to the integrated device manufacturers, fabrication houses, mask houses, foundries, and distributors.

Integrated Circuit Criticality

A variety of IC chips are designed and deployed in all mission-assurance category defense systems. Some ICs, especially in the MAC-I type systems, are custom-designed ASICs and are classified. However, not all critical ICs are classified; some ICs are unclassified but contain sensitive information. Some ICs are COTS that are widely available in civilian applications, such as dynamic random-access memory chips. For these reasons, appropriate trust levels must be placed on various classes of ICs.

Handling of classified custom ICs. Handling of classified materials and data are subject to Defense Security Services (DSS) regulations, except for those that are superseded by Government Contracting Activity. Because chip design, fabrication, testing, and packaging involve many steps, some may not be classified. In the case of wafer processing, processing steps involve iterations of mask exposure and etching. Some of the processing may not be classified, while other steps involve classified information. However, for classified wafer fabrication, appropriate DSS physical and personnel security measures must be in place and designated tools, databases, and processes, from design through finished wafers, should be considered classified as per device/system requirements.

All classified materials and data are to be managed and handled in accordance with DSS regulations. A good practice is to have a "chain of custody" throughout the manufacturing process.

In a case in which the trusted supplier may be processing classified and unclassified wafers on the same fabrication line, the facility and computers, including data storage devices—at least the part handling the classified materials and data—must be cleared to the appropriate level (e.g., Secret). Similarly, those engineers and technicians handling classified materials and data must have security clearance. Because of this mixing of classified and unclassified material and data at the same facility, as well as the mixing of cleared and uncleared personnel, a system must be in place to handle the classified materials and data properly. Furthermore, the supplier must maintain confidentiality of the customer to prevent disclosure of the end application of the fabricated chips.

Handling of sensitive but unclassified custom ICs. Some ICs, even though not classified, contain sensitive information and must satisfy a certain level of trust. These chips can be custom designed ASICs, MOTS, or COTS chips. COTS chips, for example, though widely available in the civilian sector, may have their functions altered by software and may become sensitive in unique defense applications. SBU chips require special and careful handling to ensure trust.

Certification Process

The certification process may vary for the three levels of criticality listed above. DoD will have a Certification Authority (CA), with the assistance of DSS, carry out the certification process as required. This may consist of on-site evaluation of the fabrication and mask-making operations of the potential suppliers for critical custom ICs. The inspection would include an assessment of the suppliers' operating procedures, quality management and control programs, protection of intellectual properties, and design data. Where appropriate, the certification team will apply the trusted IC supplier criteria, and if it finds the supplier can meet the criteria, will issue a Trusted Supplier Certification. The certification would generally be good for three years unless there is a significant change in the status of the supplier, such as being bought out by a foreign entity. Periodically, CA will conduct surveillance reviews to ensure the supplier continues to be qualified for certification. Recertification is required at the 3-year interval.

4.4.3.6 Additional information

A number of organizations within the U.S. federal government can provide relevant data. Specific organizations include Defense Security Service (DSS) and Defense Contract

Management Agency (DCMA). Leveraging all-source information, particularly for highly critical elements and systems, can help manage supplier risks.

For more information, see the Acquisition section of the Agreement process in section 3.1, and section 4.2, Program Protection Implementation.

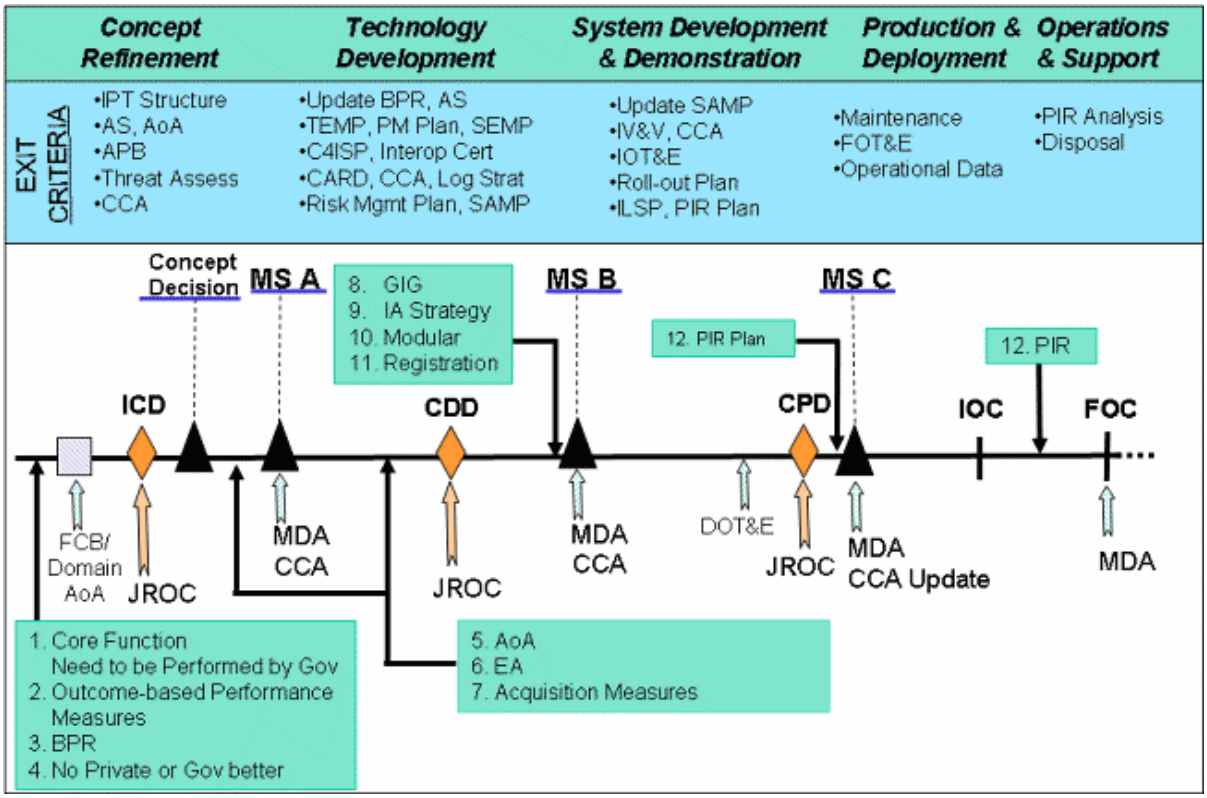
Some documents discussing supplier assurance include:

- "Mission Impact of Foreign Influence on DoD Software," Defense Science Board (DSB) Task Force on Mission Impact of Foreign Influence on DoD Software (2007)
- "High Performance Microchip Supply," DSB Task Force on High Performance Microchip Supply (2005)
- "Framework for Life cycle Risk Mitigation for National Security Systems in the Era of Globalization," Center for National Security Systems (CNSS), Global Information Technology Working Group (2006)
- "Protecting Sensitive Compartmented Information Within Information Systems," DCID (1999)
- "Defense Acquisitions. Stronger Management Practices are Needed to Improve DoD's Software-Intensive Weapons Acquisitions" General Accountability Office (GAO 2004)
- "Defense Acquisitions: Knowledge of Software Suppliers Needed to Manage Risks" (GAO 2004)
- "Cyber Security: A Crisis of Prioritization," U.S. President's Information Technology Advisory Committee (2005)
- "Mandatory Procedures for Research and Technology Protection Within the DoD" (DoD 5200.39-R)

4.5 Title 40 U.S.C. in the JCIDS and Acquisition Process

Title 40 U.S.C. (formerly, the Clinger-Cohen Act), a U.S. law, establishes that government information technology (IT) should be managed the same way as an efficient and profitable business would manage IT. Acquisition, planning, and management of technology must be treated as a "capital investment." All facets of capital planning are to be taken into consideration as they would be in private industry: cost-benefit ratio, expected life of the technology, flexibility, and possibilities for multiple uses.

Tasks for Title 40 U.S.C. span both the JCIDS and the acquisition process. Figure 4-3 shows where CCA fits within these two processes. In the chart, the exit criteria items in blue are representative samples of items that occur in each of the acquisition phases.



Source: Department of Defense

Figure 4-3 Title 40.U.S.C. (Formerly, the Clinger-Cohen Act) in the JCIDS and Acquisition Process

See Appendix C for more information about Title 40.U.S.C. (formerly, the Clinger-Cohen Act) and its relationship to system assurance.

APPENDIXES

This page intentionally left blank.

Appendix A: System Assurance Background Information

Several good references provide additional software assurance background information:

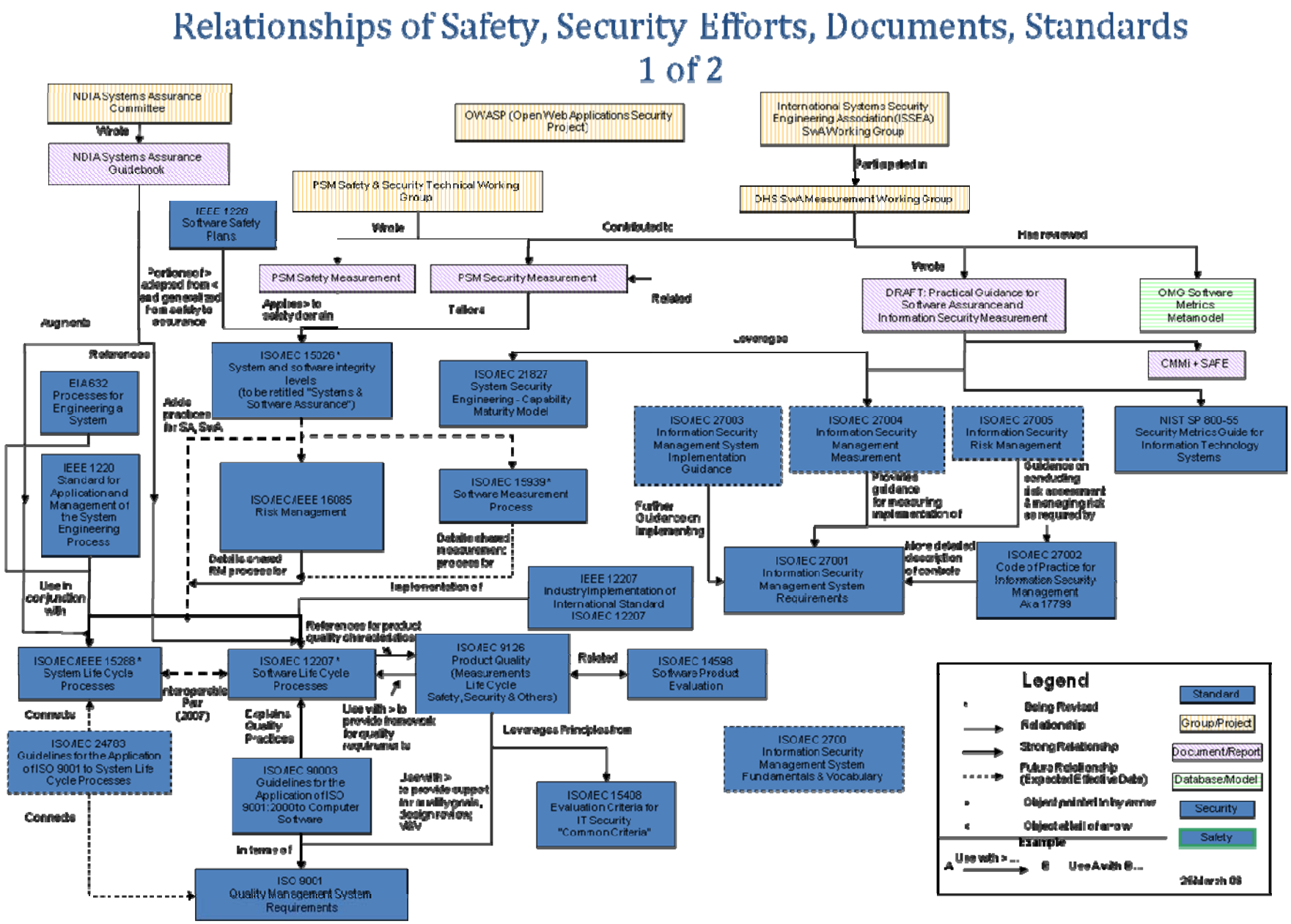
- Software Assurance: A Guide to the Common Body of Knowledge to Produce, Acquire, and Sustain Secure Software (Redwine 2006)
- Software Security Assurance: State-of-the-Art Report (Goertzel 2007)
- Security in the Software Lifecycle: Making Software Development Processes—and the Software Produced by Them—More Secure (Goertzel et al. 2006)

Redwine 2006a provides an overview of practices in the field as well as pointers to more detailed information. The current version is somewhat out-of-date but is scheduled to be revised in late 2007. Goertzel 2007 provides an assessment of the state-of-the-art for software security assurance as of July, 31, 2007, for the software development life cycle, and it describes the major efforts ongoing in this field. Goertzel 2006 describes methodologies, life-cycle processes, practices, and technologies that help improve software security.

Software assurance began as a field in the 1960s, but there was a lack of interest during the 1990s and the early 2000s (Redwine 2006). Currently, there is significant interest. Unfortunately, similar reference material is not readily available for system assurance.

Figure A-1 shows some relevant Safety and Security Efforts, Documents, and Standards. This represents a snapshot of the field taken in August 2007.

The purpose of these figures is to show the relationships between various assurance documents and efforts.



A- 1 Safety and Security Efforts, Documents, and Standards

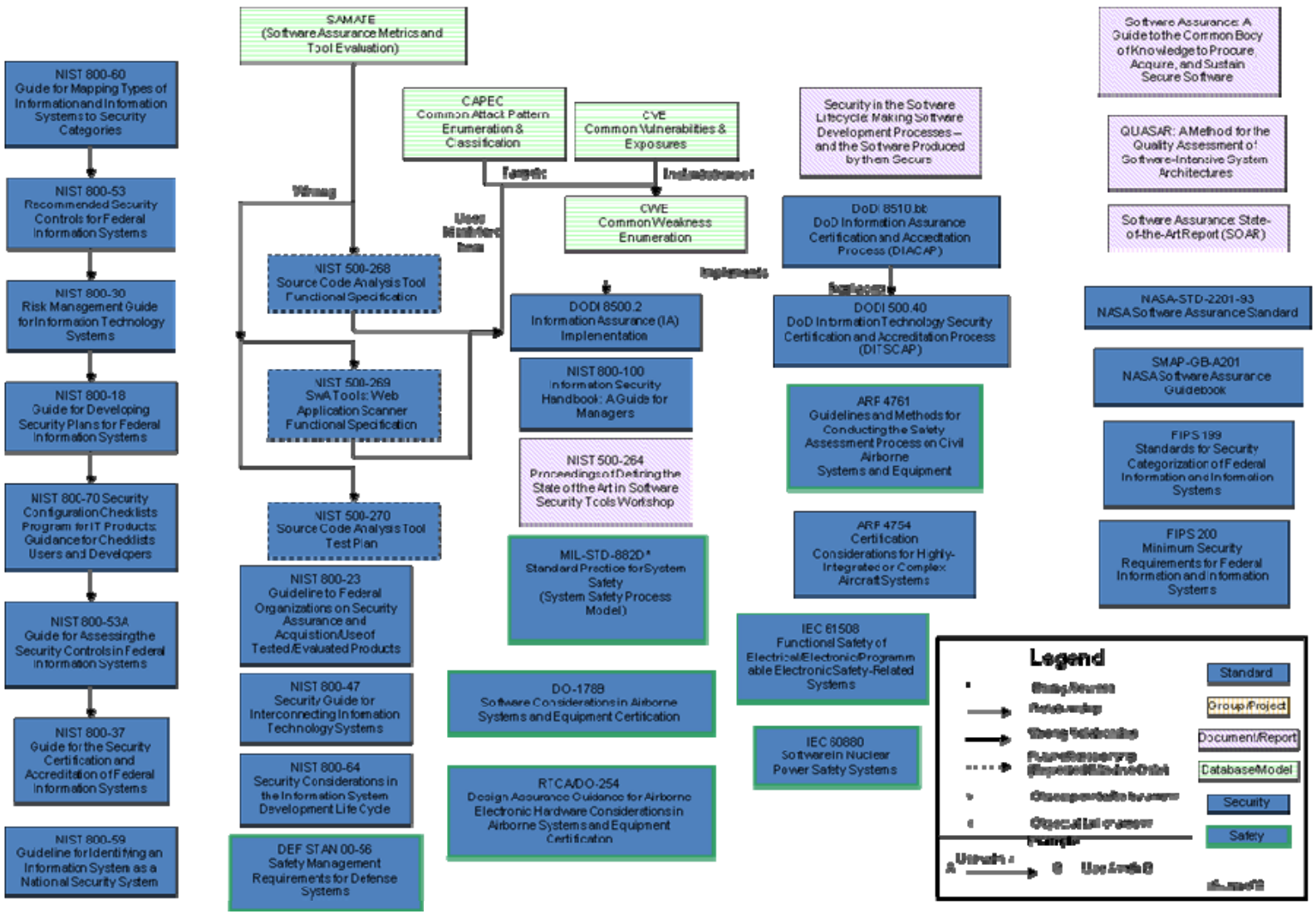


Figure A-1 Safety and Security Efforts, Documents, and Standards (continued)

Appendix B: Correspondence with Existing Documentation, Policies, and Standards

In developing assured systems, we are presented with a wide choice of laws, policies, standards, and best practices to support the processes. This appendix provides a high-level view of selected documentation and the relationship to the system assurance goals.

B.1 PUBLIC LAW

B.1.1 Computer Security Act of 1987, Public Law 100-235

A central computer security program must address compliance with national policies and requirements, as well as requirements of DoD directives, service regulations, and organization-specific requirements. National requirements include those prescribed under the Computer Security Act of 1987.

Congress declared that improving the security and privacy of sensitive information in federal computer systems is in the public interest, and created a means for establishing minimum acceptable security practices for those systems without limiting the scope of security measures already planned or in use.

- Develops standards and guidelines to ensure:
  - Cost-effective security.
    The Computer Security Act assigned to the National Bureau of Standards responsibility for developing standards and guidelines for federal computer systems, including responsibility for developing standards and guidelines needed to assure the cost-effective security of federal computer systems.
  - Privacy of sensitive information.
    This act also assigned to the National Bureau of Standards responsibility for developing standards for the privacy of sensitive information in federal computer systems and drawing on the technical advice and assistance (including work products) of the National Security Agency where appropriate.
- Provides for standards and guidelines.
  The law requires the development of standards and guidelines for federal computer systems needed to assure the cost-effective security and privacy of sensitive information in federal computer systems.
- Requires security plans.
  The Computer Security Act provides for the establishment of security plans by all operators of federal computer systems that contain sensitive information. It also requires mandatory periodic training for all persons involved in management, use, or operation of federal computer systems that contain sensitive information.
- Requires mandatory periodic training.
  The Computer Security Act states that “each federal agency shall provide for the mandatory periodic training in computer security awareness and accepted computer practices of all employees who are involved with the management, use, or operation of each federal computer system within or under the supervision of that agency.”

B.1.2 The Privacy Act of 1974, 5 U.S.C. § 552

The Privacy Act of 1974, 5 U.S.C. § 552a (1994 & Supp. IV 1998) became effective on September 27, 1975. The historical context of the Privacy Act is important to understanding its remedial purposes: In 1974, Congress was concerned with curbing the illegal surveillance and investigation of individuals by federal agencies that had been exposed during the Watergate scandal; it was also concerned with potential abuses presented by the government's increasing use of computers to store and retrieve personal data by means of a universal identifier—such as an individual's social security number.

The Act focuses on four basic policy objectives:

- To restrict disclosure of personally identifiable records maintained by agencies.
- To grant individuals increased rights of access to agency records maintained on them.
- To grant individuals the right to seek amendment of agency records maintained on themselves upon a showing that the records are not accurate, relevant, timely, or complete.
- To establish a code of "fair information practices" that requires agencies to comply with statutory norms for collection, maintenance, and dissemination of records.

B.1.3 OMB Circular A-130, Management of Federal Information Resources

The federal government is the largest single producer, collector, consumer, and disseminator of information in the United States. Federal agencies, state and local governments, and the public are dependent on government information resources. Other entities dependent on this information are local government agencies, educational, and other not-for-profit institutions, and for-profit organizations. Hence, the OMB views government information systems as a valuable national resource. It provides the public with knowledge of the government, society, and economy—past, present, and future. It is a means to ensure the accountability of government, to manage the government's operations, and to maintain the healthy performance of the economy and is itself a commodity in the marketplace.

The OMB also recognizes that the free flow of information between the government and the public is essential to a democratic society. At the same time, it is also essential that the government minimize the federal paperwork burden on the public, minimize the cost of its information activities, and maximize the usefulness of government information.

Because the public disclosure of government information is essential to the operation of a democracy, the management of federal information resources should protect the public's right of access to government information.

A central security program should provide two quite distinct types of benefits:

- Increased efficiency and economy of security throughout the organization
- The ability to provide centralized enforcement and oversight

Both of these benefits are in keeping with the purpose of the Paperwork Reduction Act as implemented in OMB Circular A-130.

OMB Circular A-130 Requires:

- Information security plans
- Computer security plans
- Awareness training

- Contingency planning
- Formal emergency response capabilities
- Protection of Privacy Act information
- Cooperation between federal, state, and local government

B.1.4 Federal Information Security Management Act of 2002 (FISMA) (Title III of the E-Gov Act of 2002)

Title III contains six sections:

- Section 301 revises GISRA.
- Section 302 amends the federal information systems standards provisions of the Clinger-Cohen Act regarding promulgation of NIST-developed information security standards.
- Section 303 revises NIST's information security and systems standards role.
- Section 304 revises the role of the NIST Computer System Security and Privacy Advisory Board.
- Section 305 makes technical and conforming amendments, including repealing unnecessary provisions of the Computer Security Act and the Paperwork Reduction Act.
- Section 306 provides rules of construction to ensure proper implementation of standards promulgation authority by OMB and the Department of Commerce.

Section 301 (Information Security) above provides the following:

- Provides a comprehensive framework for ensuring the effectiveness of information security through a risk-based entity-wide management approach;
- Recognizes the highly networked nature of the federal computing environment and provides effective government-wide management and oversight of related information security risks;
- Provides for the development and maintenance of minimum security controls;
- Provides for improved oversight through annual independent evaluations of agency information security practices;
- Acknowledges that COTS security products offer effective information solutions; and
- Leaves the selection of specific COTS security solutions to individual agencies.

B.1.5 Electronic Government Act of 2002 (E-Gov), December 17, 2002

Provides means for improving coordination and deployment of IT across the federal government, helping agencies achieve the IT management reforms required under the Clinger-Cohen Act, and ensuring greater citizen access to the federal government through the improved use of IT. Establishes an Office of Electronic Government within OMB and an E-Government Fund at $345 million over fiscal years 2003 through 2006.

B.1.6 Clinger-Cohen Act (formerly the Information Technology Management Reform Act of 1996), February 10, 1996

Combines the Information Technology Management Reform Act and the Federal Acquisition Reform Act. Gives federal agencies full responsibility and authority for the acquisition and management of their IT resources, including technology used by the agency directly or used by a contractor under contract to the agency. Requires each agency to implement processes for

maximizing the value and managing the risks of the agency's IT acquisitions. Requires the appointment of a CIO for the 24 largest Federal agencies, and assigns specific duties to CIOs.

B.1.7 Other Public Laws

United States Public Law 104-106, Information Technology Act of 1996.

B.2 Executive Policy (White House, Department)

Critical Infrastructure Assurance Office, *Practices for Securing Critical Information Assets*, January 2000.

Executive Office, Homeland Security Presidential Directive 7, *Critical Infrastructure Identification, Prioritization, and Protection*, December 17, 2003.

Executive Office, Homeland Security Presidential Directive 12, *Policy for a Common Identification Standard for Federal Employees and Contractors*, August 27, 2004.

HSPD-7, Critical Infrastructure Identification, Prioritization, and Protection, December 17, 2003. This directive establishes a national policy for federal departments and agencies to identify and prioritize the United States' critical infrastructure and key resources and to protect them from terrorist attacks.

EO 13231, Critical Infrastructure Protection in the Information Age, 16 Oct 2001. Establishes policy to ensure protection of information systems for critical infrastructure, including emergency preparedness communications, and the physical assets that support such systems.

Executive Order 12958, as amended, *Classified National Security Information*, April 17, 1995.

EO 13011, Federal Information Technology, 16 Jul 96. Establishes policy that agencies will: 1) significantly improve management of information systems; 2) refocus IT management to support directly strategic missions; 3) establish clear accountability for information resources management activities by creating CIOs; 4) cooperate in the use of IT to improve the productivity of federal programs; and 5) establish interagency support. Also establishes CIO Council, comprised of CIOs and Deputy CIOs.

B.3 Services Standards

B.3.1 Department of Defense Directive 5000.1, The Defense Acquisition System, December 5, 2003

This Directive:

- Reissues reference (a) and authorizes publication of reference (b).
- Along with reference (b), provides management principles and mandatory policies and procedures for managing all acquisition programs.

B.3.2 Department of Defense Instruction 5000.2, Operation of the Defense Acquisition System, May 12, 2003

This Instruction: (1) Establishes a simplified and flexible management framework for translating mission needs and technology opportunities, based on approved mission needs and requirements, into stable, affordable, and well-managed acquisition programs that include weapon systems and automated information systems (AISs); (2) Consistent with statutory requirements and reference

(c), authorizes Milestone Decision Authorities (MDAs) to tailor procedures to achieve cost, schedule, and performance goals.

B.3.3 Department of Defense Directive 8500.1, Information Assurance, October 24, 2002

This Directive establishes policy and assigns responsibilities under Section 2224 of Title 10, United States Code, to achieve Department of Defense (DoD) information assurance (IA) through a defense-in-depth approach that integrates the capabilities of personnel, operations, and technology, and supports the evolution to network-centric warfare.

B.3.4 Department of Defense Instruction 8500.2, Information Assurance Implementation, February 6, 2003

This Instruction implements policy, assigns responsibilities, and prescribes procedures for applying integrated, layered protection of the DoD information systems and networks under DoD Directive 8500.1.

B.3.5 Department of Defense Directive 8570.1, Information Assurance Training, Certification and Workforce Management

This Directive establishes policy and assigns responsibilities, according to DoD Directive 8500.1, "Information Assurance," October 24, 2002; DoD Instruction 8500.2, "Information Assurance (IA) Implementation," February 6, 2003; DoD Directive O-8530.1, "Computer Network Defense," January 8, 2001; Section 3544 of Title 44 United States Code for DoD IA training, certification, and workforce management.

B.3.6 Department of Defense Instruction 8580.1, Information Assurance in the Defense Acquisition System, July 9, 2004

This Instruction: 1) Implements policy, assigns responsibilities, and prescribes procedures under Chapter 25 of Title 40, United States Code, DoD Directive 8500.1, "Information Assurance," October 24, 2002, and DoD Instruction 8500.2, "Information Assurance (IA) Implementation," February 6, 2003 necessary to integrate IA into the Defense Acquisition System described in DoD Directive 5000.1, "The Defense Acquisition System," May 12, 2003 and DoD Instruction 5000.2, "Operation of the Defense Acquisition System," May 12, 2003; 2) Describes required and recommended levels of IA activities relative to the acquisition of systems and services: 3) Describes the essential elements of an Acquisition IA Strategy and its applicability, and prescribes an Acquisition IA Strategy submission and review process.

B.3.7 Chairman of the Joint Chiefs of Staff Instruction (CJCSI) 6512.01D, *Interoperability and Supportability of Information Technology and National Security Systems, March 8, 2006*

This Instruction establishes policies and procedures for developing, coordinating, reviewing, and approving IT and National Security System (NSS) Interoperability and Supportability (I&S) needs.

B.3.8 Chairman of the Joint Chiefs of Staff Instruction (CJCSI) 6510.01E, *Information Assurance (IA) and Computer Network Defense (CND), August 15, 2007*

This instruction provides joint policy and guidance for information assurance and computer network defense.

B.3.9 Army Regulation 25-2 Information Assurance

The protection of an information system includes preserving the integrity, availability, and confidentiality of information system resources (including hardware, software, firmware, information/data, and telecommunications). This regulation mandates that agencies protect their computers, the information they process, and related technology resources. In relationship to system assurance, AR 25-2:

- Provides administrative and systems security requirements, including those for interconnected systems.
- Promotes the use of efficient procedures and cost-effective, computer-based security features and assurances.
- Requires a life-cycle management approach to implementing IA requirements.
- Requires the implementation of a Configuration Management Process.
- Describes the Continuity of Operations Plan (COOP).
- Establishes and implements the concept of Best Business Practices (BBPs) to define and enforce mandatory and specific measures, practices, or procedures to meet changing technology or IA requirements.
- Provides the foundation for the Networthiness Certification Program.

B.3.10 Secretary of Navy Instruction 5239.3A, Department of the Navy IA Program, December 2004

This instruction establishes within the Department of the Navy an IA policy that provides information security protections commensurate with the risk and magnitude of the harm resulting from unauthorized access.

B.3.11 Chief of Naval Operations Instruction 5239.1B, Navy IA Program, November 9, 1999

This instruction establishes policies and procedures for the U.S. Navy's IA Program.

B.3.12 SECNAV M-5239.1, Department of Navy IA Manual

This Manual implements the policy set forth in Secretary of the Navy Instruction (SECNAVINST) 5239.3A, Subject: "Department of the Navy Information Assurance Policy" and is issued under the authority of SECNAVINST 5430.7N, Subject: "Assignment of Responsibilities and Authorities in the Office of the Secretary of the Navy." This Manual is intended to serve as a high-level introduction to IA and IA principles. It discusses common IA controls and associated requirements and reviews the DoD strategy for implementing those controls.

B.3.13 U.S. Marine Corps Information Assurance Program, November 18, 2002

This MCO formally establishes the MCIAP and defines the responsibilities for protecting our information infrastructure.

B.3.14 AFPD 33-2, Information Assurance (IA) Program, April 19, 2007

http://www.e-publishing.af.mil/shared/media/epubs/AFPD33-2.pdf

B.3.15 AFI 33-202, Network and Computer Security, Vol 1, February 3, 2006, Incorporating Through Change 5, May 18, 2007

http://www.e-publishing.af.mil/shared/media/epubs/AFI33-202V1.pdf

B.3.16 DoD 5200.1-M, Acquisition Systems Protection Program, March 1994 [http://www.dtic.mil/whs/directives/corres/pdf/520001m.pdf]

This manual prescribes standards, criteria, and methodology for the identification and protection of DoD critical program information (CPI) within DoD acquisition programs. The standards and criteria in the manual are intended to protect against loss and unauthorized disclosure of CPI throughout the acquisition process at all involved locations or facilities. They will also identify and reduce projected operational system susceptibility to damage, compromise, or destruction. The ultimate goal is to selectively and effectively apply security countermeasures to protect the CPI and reduce costs by applying risk management.

B.3.17 DoDD 5200.39, Security, Intelligence, and Counterintelligence Support to Acquisition Program Protection, September 19, 1997 [http://www.dtic.mil/whs/directives/corres/pdf/520039p.pdf]

This document establishes policy and assigns responsibilities to security, intelligence, and counterintelligence (CI) activities that provide support to acquisition organizations after CPI have been identified.

B.4 NIST/NSA (NIAP) STANDARDS

B.4.1 National Institute of Standards and Technology

The U.S. National Institute of Standards and Technology (NIST) produces a wide range of technical standards and (unlike ISO) they are available to all, free of charge. The standards are intended for U.S. Government, military, and commercial use. Here is a selection of NIST Special Publication 800-series standards that are relevant to information security management systems and ISO 27000 (SP 800-53, for one, is well worth a look):

- SP 800-12 (Oct 1995) *An Introduction to Computer Security: The NIST Handbook* may be getting a bit outdated but serves as a general introduction to, for example, security policies and procedures.
- SP 800-18 (Dec 1998, Rev 1, Feb 2006) *Guide for Developing Security Plans for Information Technology Systems* guides the design and documentation of IT security controls.
- SP 800-23 (Aug 2000) *Guideline to Federal Organizations on Security Assurance and Acquisition/Use of Tested/Evaluated Products*
- SP 800-27 (Jun 2004 revision A) *Engineering Principles for Information Technology Security (A Baseline for Achieving Security).*
- SP 800-28 (March 2008) *Guidelines on Active Content and Mobile Code*.
- SP 800-30 (July 2002) *Risk Management Guide for Information Technology Systems* guides the assessment and control of IT risks.
- SP 800-34 (June 2002) *Contingency Planning Guide for Information Technology Systems*.
- SP 800-35 (Oct 2003) *Guide to Information Technology Security Services*.
- SP 800-36 (Oct 2003) *Guide to Selecting Information Security Products*.

- SP 800-37 (May 2004) *Guide for the Security Certification and Accreditation of Federal Information Systems* provides guidance on security certification, accreditation, and authorization of information systems.
- SP 800-39 (Oct 2007) *Managing Risk from Information Systems, and Organizational Perspective.*
- SP 800-40 (Nov 2005, V 2) *Creating a Patch and Vulnerability Management Program* (formerly *Procedures for Handling Security Patches*).
- SP 800-42 (Oct 2003) *Guideline on Network Security Testing.*
- SP 800-45 (Version 2 February 2007) *Guidelines on Electronic Mail Security*.
- SP 800-46 (Aug 2002) *Security for Telecommuting and Broadband Communications*.
- SP 800-47 (Aug 2002) *Security Guide for Interconnecting Information Technology Systems*.
- SP 800-48 (Nov 2002) *Wireless Network Security: 802.11, Bluetooth, and Handheld Devices*.
- SP 800-50 (Oct 2003) *Building an Information Technology Security Awareness and Training Program* is recommended reading for anyone planning a professional approach to security awareness, training, and education activities.
- SP 800-53 (Feb 2005) *Recommended Security Controls for Federal Information Systems*, in effect another ISMS standard, contains a handy cross-reference table comparing its control coverage to that of standards such as ISO 17799:2005, SP800-26, Government Audit Office Federal Information System Controls Audit Manual, Department of Defense Instruction 8500.2, Information Assurance *Implementation* and Director of Central Intelligence Directive (DCID) 6/3 Policy and Manual *Protecting Sensitive Compartmented Information within Information Systems*. It explains the process of implementing and then building on a security baseline. Includes Draft SP 800-53A, *Guide for Assessing the Security Controls in Federal Information Systems.*
- SP 800-55 (July 2003) *Security Metrics Guide for Information Technology Systems* sounds more useful than it is; it is essentially a long list of security items that "could be measured."
- SP 800-56A (Mar 2006) *Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography*.
- SP 800-57 Part 1 and Part 2 (Aug 2005) *Recommendation on Key Management.*
- SP 800-58 (Jan 2005) *Security Considerations for Voice Over IP Systems.*
- SP 800-59 (Aug 2003) *Guideline for Identifying an Information System as a National Security System* provides guidance on identifying an information system as a (US) national security system, presumably implying the need for strict security controls (although in fact there is a large overlap between commercial and defense security requirements, hence the move to accept ordinary, cheaper commercial IT solutions for military use where appropriate).
- SP 800-60 and appendices (Jun 2004) *Guide for Mapping Types of Information and Information Systems to Security Categories* guides the assignment of security categories to information systems.
- SP 800-61 (Revision 1 March 2008) *Computer Security Incident Handling Guide*.
- SP 800-63 (Sep 2004) *Electronic Authentication Guideline*.
- SP 800-64 (Jun 2004) *Security Considerations in the Information System Development Life Cycle*.

- SP 800-65 (Jan 2005) *Integrating Security into the Capital Planning and Investment Control Process*.
- SP 800-70 (May 2005) *Security Configuration Checklists Program for IT Products: Guidance for Checklists Users and Developers* comprises a set of baseline configurations for a wide variety of operating system platforms.
- SP 800-80 (May 2006 DRAFT) *Guide for Developing Performance Metrics for Information Security.*
- SP 800-83 (Nov 2005) *Guide to Malware Incident Prevention and Handling.*
- SP 800-88 (Sep 2006) *Guidelines for Media Sanitization Contains* vital information if an organization has unsanitary computer media with viruses.
- SP 800-92 (Sep 2006) *Guide to Computer Security Log Management.*
- SP 800-100 (Jun 2006 DRAFT) *Information Security Handbook: a Guide for Managers.*

Like ISO and BSI, NIST has published various other security-related standards over the years in addition to those in the SP 800 series, including Federal Information Processing Standards (FIPS) Publication standards such as:

- FIPS 199 (Feb 2004) Standards for Security Categorization of Federal Information and Information Systems.
- FIPS 200 (Mar 2006) Minimum Security Requirements for Federal Information and Information Systems.
- FIPS 201 (Feb 2005) *Personal Identity Verification for Federal Employees and Contractors.*

NIST, NSA and DISA have jointly developed/released several Security Technical Implementation Guides (STIGs) and related documents. These form an excellent basis for corporate security standards.

B.4.2 Committee on National Security Systems (CNSS) (formerly National Security Telecommunications and Information Systems Security Committee (NSTISSC))

- National Security Telecommunications and Information Systems Security Committee (NSTISSC), *National Information Assurance Certification and Accreditation Process (NIACAP)*, (Washington, DC, April 2000). Note: NSTISSC is now the Committee on National Security Systems (CNSS). NSTISSI No. 1000.
- National Security Telecommunications and Information Systems Security Committee, *National Information Systems Security Glossary*, (Washington, DC, September 2000). NSTISSI No. 4009.

B.5 COMMERCIAL STANDARDS–IEEE, ISO

B.5.1 Institute of Electrical and Electronics Engineers (IEEE) Standards

The IEEE is a leading developer of standards that underpin many of today's technologies. The standards are developed in a unique environment that builds consensus in an open process based on input from all interested parties. With nearly 1,300 standards either completed or under development, it is a central source of standardization in both traditional and emerging fields, particularly telecommunications, information technology, and power generation.

The IEEE also develops standard jointly with ISO/IEC JTC1/SC7, the subcommittee for systems and software engineering standards.

Some of the standards relevant to the system assurance process include:

- ISO/IEC/IEEE 12207-2008, Systems and software engineering — Software life cycle processes.

  This standard establishes a common framework for software life cycle processes, with well-defined terminology, that can be referenced by the software industry. It contains processes, activities, and tasks that are to be applied during the acquisition of a software product or service and during the supply, development, operation, maintenance and disposal of software products.

- ISO/IEC/IEEE 15288-2008, Systems and software engineering — System life cycle processes.

  This standard establishes a common framework for describing the life cycle of systems created by humans. It defines a set of processes and associated terminology. These processes can be applied at any level in the hierarchy of a system's structure. Selected sets of these processes can be applied throughout the life cycle for managing and performing the stages of a system's life cycle.

- ISO/IEC/IEEE 14764-2006, Software Engineering — Software Life Cycle Processes — Maintenance.

  This standard defines the process for performing the maintenance of software.

- ISO/IEC/IEEE 16085-2006, Systems and software engineering — Life cycle processes — Risk management.

  This standard describes a process for the management of risk during systems or software acquisition, supply, development, operations, and maintenance.

- IEEE Std 610.12-1990, Standard Glossary of Software Engineering Terminology

  This standard contains definitions for more than 1,000 terms, establishing the basic vocabulary of software engineering. [This standard is being replaced by ISO/IEC IEEE 24765, Systems and software engineering — Vocabulary, and by the SEVOCAB terminology database

- IEEE Std 1233-1998, Guide for Developing of System Requirements Specifications

  This standard provides guidance for the development of a set of requirements that, when realized, will satisfy an expressed need.

- IEEE Std 1058-1998, Standard for Software Project Management Plans

  This standard specifies the format and contents of software project management plans.

- IEEE Std 1074-2006, IEEE Standard for Developing a Software Project Life Cycle Process

  This standard provides a process for creating a software project life cycle process (SPLCP). It is primarily directed at the process architect for a given software project. It is the function of the process architect to develop the SPLCP.

- IEEE Std 730.1-1998, Guide for Software Quality Assurance Plans

  The purpose of this guide is to identify approaches to good Software Quality Assurance practices in support of IEEE Std 730. (The standard establishes a required format and a set of minimum contents for Software Quality Assurance Plans. The description of each of the required elements is sparse and thus provides a template for the development of further standards, each expanding on a specific section of this document.)

- IEEE Std 1008-1987 (Reaff 1993), Standard for Software Unit Testing

  The standard describes a testing process composed of a hierarchy of phases, activities, and tasks. Further, it defines a minimum set of tasks for each activity.

- IEEE Std 1063-2001, Standard for Software User Documentation

  This standard provides minimum requirements for the structure and information content of user documentation.

B.5.2 ISO Standards

- ISO/IEC 90003-2004, Software and Systems Engineering – Guidelines for the Application of ISO 9001:2000 to Computer Software

  This standard provides guidance to organizations in the application of ISO 9001:2000 to the acquisition, supply, development, operations, and maintenance of computer software.

- ISO 13335 - IT security management

  ISO 13335 (which started life as a Technical Report (TR) before becoming a full ISO standard) comprises a set of guidelines for the management of IT security, focusing primarily on technical security control measures:

  - ISO 13335-1:2004 Information technology – Security techniques – Management of information and communications technology security – Part 1: Concepts and models for information and communications technology security management explains the concepts and models for information and communications technology security management. (ISO/IEC TR 13335 parts 1 and 2 were combined into the revised ISO/IEC 13335-1: 2004. The original TR13335-2:1997 “Guidelines for the management of IT security - Part 2: Managing and planning IT security” was cancelled.)
  - ISO 13335-2, when published, is expected to cancel and replace ISO/IEC TR 13335-3:1998 and ISO/IEC TR 13335-4:2000.
  - ISO TR 13335-3:1998 Information technology – Guidelines for the Management of IT Security – Part 3: Techniques for the management of IT Security covers techniques for the management of IT security. This standard is currently under revision and will be inserted into ISO 27005
  - ISO TR 13335-4:2000 covers the selection of safeguards (meaning technical security controls). This standard is also currently under revision and will be inserted into ISO 27005
  - ISO TR 13335-5:2001 provides management guidance on network security. This standard is currently under revision, being merged into ISO/IEC 18028-1. ISO/IEC 18028-1 will eventually cancel and replace ISO/IEC TR 13335-5:2001.ISO 15408 - Common Criteria

- ISO 15408:1999 describes the Common Criteria for Information Technology Security Evaluation. Products that are evaluated against the Common Criteria have a defined level of assurance for their information security capabilities that is recognized in most of the world. Unfortunately, the evaluation process is quite costly and slow, and is therefore not very widely used apart from the government and defense markets.

- ISO 19770 - Software Asset Management

  ISO/IEC 19770-1:2006 promotes the implementation of an integrated set of software asset management processes, using good practices for efficient software management. Contents include Scope, terms and definitions, Field of application, Conformance, Intended usage, Agreement compliance, General Software Asset Management processes, Control

environment for Software Asset Management, Planning and implementation, Inventory processes, Verification and compliance processes, Operations management processes and interfaces, Life cycle process interfaces.

- ISO 21827 - Systems Security Engineering Capability Maturity Model

  Like other Capability Maturity Models (CMMs), the Systems Security Engineering (SSE) CMM defines the essential characteristics of SSE processes, emphasizing those which indicate process maturity. The model covers the entire systems development life cycle from concept definition to decommissioning. It applies to those developing or integrating secure products/systems, and those providing specialist security services such as security engineering. It was published as ISO 21827 in 2002.
- ISO has reserved the ISO/IEC 27000-series numbering for a range of information security management standards in similar fashion to the very successful ISO 9000-series quality assurance standards.

The following ISO 27000-series standards are either published or planned:

- ISO 27000 will contain the vocabulary and definitions, i.e., terminology for all of these information security management standards
- ISO 27001 is the Information Security Management System (ISMS) requirements standard (specification) against which organizations are formally certified compliant. In October 2005, British Standard BS 7799 part 2 was adopted by ISO, rebadged and released as the new international information security standard ISO/IEC 27001:2005. ISO 27001 is the formal standard against which organizations may seek independent certification of their ISMS (meaning their frameworks to design, implement, manage, maintain, and enforce information security processes and controls systematically and consistently throughout the organizations).
- ISO 27002 will be the new name for the standard currently known as ISO 17799 and formerly known as BS 7799 part 1. This is the code of practice for information security management describing a comprehensive set of information security control objectives and a menu of best-practice security controls.
- ISO 27003 will be an implementation guide.
- ISO 27004 will be an information security management measurement standard to help measure the effectiveness of ISMS implementations.
- ISO 27005 will be an information security risk management standard (will replace the recently issued BS 7799 Part 3).
- ISO 27006 will be a guide to the certification/registration process for accredited ISMS certification/registration bodies.

B.6 BEST PRACTICE (E.G., DHS/DOD SWACBK)

B.6.1 COBIT v4

COBIT (Control Objectives in IT) from ISACA (formerly known as the IS Audit and Control Association and still known as a professional body representing IT auditors) has matured over the past decade from quite modest beginnings as a guide for computer auditors on best practice IT management controls into a comprehensive tool to guide the implementation of sound IT governance processes/systems.

The latest release, COBIT v4, is described by ISACA as “an IT governance framework and supporting toolset that allows managers to bridge the gap between control requirements, technical

issues, and business risks. COBIT enables clear policy development and good practice for IT control throughout organizations ... [It] emphasizes regulatory compliance, helps organizations to increase the value attained from IT, enables alignment, and simplifies implementation of the COBIT framework."

B.6.2 Information Security Forum (ISF) Standard of Good Practice

The ISF's Standard of Good Practice for Information Security, last updated in January 2005, has long been well regarded as a broadly scoped pragmatic standard for information security. It is available free of charge from the ISF Web site and provides a useful crosscheck on the coverage and content of security policies and procedures written to follow ISO 17799 or other standards.

B.6.3 GAISP

GAISP (Generally Accepted Information Security Practices) developed from and consolidated earlier works such as GASSP (Generally Accepted System Security Practices). At one time ISSA (the Information Systems Security Association) was reworking it, although the project has been largely overtaken by events such as the release of ISO 27000.

B.6.4 TickIT

TickIT is a software Quality Assurance (QA) framework built upon the foundations of ISO 9001 and ISO 12207 processes. Its major objective was to provide industry with a practical framework for the management of software development quality by developing more effective quality management system certification procedures. These involved: publishing guidance material to assist software organizations, to interpret the requirements of ISO 9001; training, selecting, and registering auditors with IT experience and competence; and introducing rules for the accreditation of certification bodies practicing in the software sector.

B.6.5 Information Technology Infrastructure Library (ITIL)

ITIL is a framework of best practice approaches intended to facilitate the delivery of high-quality IT services. ITIL outlines an extensive set of management procedures that are intended to support businesses in achieving both quality and value for money in IT operations. These procedures are supplier-independent and have been developed to provide guidance across the breadth of IT infrastructure, development, and operations. Specifically, the framework includes CM, service-level management, application management, security management, and software asset management.

Appendix C: DoD Mappings

This appendix provides mappings from:

- DoD IA controls to guidebook material and the DoD Life cycle
- USC Title 40 (formerly called the Clinger-Cohen Act) to System Assurance requirements.

C.1 DoD IA Mappings

C.1.1 Mapping DoD controls (DoDI 8500.2) to guidebook material

Table 4-3 shows the mappings, from relevant DoD information assurance (IA) controls that are documented in DoD Instruction 8500.2 within enclosure 4 and attachments 1 and 4, to the ISO life cycle processes as well as the DAG review cycles.

The most stringent mission assurance category and confidentiality levels, and their associated information assurance controls, were used, so that the most comprehensive list of controls is included. Note the mapping is easier to review with the DODI 8500.2 enclosure 4 at hand as Table 4-3 only has the IA control acronyms and a top-level description, and not the detail.

Table C-1 DoD Controls (DoDI 8500.2) and SA Guidebook

IA Control from 8500.2 (A1 or A4)	
DCAR-1	Procedural Review
DCBP-1	Best Security Practices
DCCB-2	Control Board
DCCS-2	Configuration Specifications
DCCT-1	Compliance Testing
DCDS-1	Dedicated IA Services
DCFA-1	Functional Architecture for AIS Applications
DCHW-1	HW Baseline
DCID-1	Interconnection Documentation
DCII-1 IA	Impact Assessment
DCIT-1	IA for IT Services
DCMC-1	Mobile Code
DCNR-1	Non-repudiation
DCPA-1	Partitioning the Application
DCPB-1	IA Program and Budget
DCPD-1	Public Domain Software Controls
DCPP-1	Ports, Protocols, and Services
DCPR-1	CM Process
DCSD-1	IA Documentation
DCSL-1	System Library Management Controls
DCSP-1	Security Support Structure Partitioning
DCSQ-1	Software Quality
DCSS-2	System State Changes
DCSW-1	SW Baseline
IAKM-2	Key Management
IATS-2	Token and Certificate Standards
ECAT-2	Audit Trail, Monitoring, Analysis, and Reporting
ECCD-2	Changes to Data
ECDC-1	Data Change Controls

IA Control from 8500.2 (A1 or A4)	
ECID-1	Host Based IDS
ECIM-1	Instant Messaging
ECND-2	Network Device Controls
ECPA-1	Privileged Account Control
ECPC-2	Production Code Change Controls
ECRG-1	Audit Reduction and Report Generation
ECSC-1	Security Configuration Compliance
ECSD-2	Software Development Change Controls
ECTB-1	Audit Trail Backup
ECTM-2	Transmission Integrity Controls
ECTP-1	Audit Trail Protection
ECVI-1	Voice over IP
ECVP-1	Virus Protection
ECWN-1	Wireless Computing and Networking
EBCR-1	Connection Rules
EBVC-1	VPN Controls
PEEL-2	Emergency Lighting
PEFD-2	Fire Detection
PEFI-1	Fire Inspection
PEFS-2	Fire Suppression System
PEHC-2	Humidity Controls
PEMS-1	Master Power Switch
PESL-1	Screen Lock
PETC-2	Temperature Controls
PETN-1	Environmental Control Training
PEVR-1	Voltage Regulators
PRRB-1	Security Rules of Behavior or Acceptable Use Policy
COAS-2	Alternate Site Designation
COBR-1	Protection of Backup and Restoration Assets
CODB-3	Data Backup Procedures
CODP-3	Disaster and Recovery Planning
COEB-2	Enclave Boundary Defense
COED-2	Scheduled Exercises and Drills
COEF-2	Identification of Essential Functions
COMS-2	Maintenance Support
COPS-3	Power Supply
COSP-2	Spares and Parts
COSW-1	Backup Copies of Critical SW
COTR-1	Trusted Recovery
VIIR-2	Incident Response Planning
VIVM-1	Vulnerability Management
DCAS-1	Acquisition Standards
DCSR-3	Specified Robustness – High
DCSS-2	System State Changes
IAGA-1	Group Identification and Authentication
IAIA-2	Individual Identification and Authentication
IAKM-3	Key Management
ECAD-1	Affiliation Display
ECAN-1	Access for Need-to-Know

IA Control from 8500.2 (A1 or A4)	
ECAR-3	Audit Record Content
ECAT-2	Audit Trail, Monitoring, Analysis and Reporting
ECCD-2	Changes to Data
ECCM-1	COMSEC
ECCR-2	Encryption for Confidentiality (Data at Rest)
ECCR-3	Encryption for Confidentiality (Data at Rest)
ECCT-2	Encryption for Confidentiality (Data in Transit)
ECIC-1	Interconnections among DoD Systems and Enclaves
ECLC-1	Audit of Security Label Changes
ECLO-2	Logon
ECLP-1	Least Privilege
ECML-1	Marking and Labeling
ECMT-2	Conformance Monitoring and Testing
ECNK-1	Encryption for Need-To-Know
ECNK-2	Encryption for Need-To-Know
ECRC-1	Resource Control
ECRR-1	Audit Record Retention
ECTB-1	Audit Trail Backup
ECTC-1	Tempest Controls
ECWM-1	Warning Message
IAAC-1	Account Control
EBBD-3	Boundary Defense
EBRP-1	Remote Access for Privileged Functions
EBRU-1	Remote Access for User Functions
PECF-2	Access to Computing Facilities
PECS-2	Clearing and Sanitizing
PEDD-1	Destruction
PEDI-1	Data Interception
PEPF-2	Physical Protection of Facilities
PEPS-1	Physical Security Testing
PESP-1	Workplace Security Procedures
PESS-1	Storage
PEVC-1	Visitor Control to Computing Facilities
PRAS-2	Access to Information
PRMP-2	Maintenance Personnel
PRNK-1	Access to Need-to-Know Information
PRTN-1	Information Assurance Training

C.1.2 Mapping DIACAP to Information Assurance Controls and the DoD Life Cycle

The activities described in this guidebook significantly aid in implementing the DoD Information Assurance Certification and Accreditation Process (DIACAP). The assurance case can help determine how to integrate IA controls into the systems engineering life cycle, by identifying which controls are needed, where, and for what purpose(s). IA controls apply to both the development and operational environments, not just the operational environment. Thus, in system development and demonstration, the controls must be in place, activated, and managed before they are used. Note that DIACAP is not mandatory for some DoD systems, but even in those cases it provides useful guidance.

The following table maps the DIACAP to especially relevant DoDI 8500.2 controls, Integrated Defense Acquisition, Technology and Logistics Life Cycle Management Framework phases, and DAG reviews. It differs from Table 4-2, as it does not take the DIACAP into consideration. Table 4-3 provides the PMs and SEs with a direct connection between the IA controls, the DAG reviews with the DIACAP life cycle process. Note the mapping is easier to review with the DODI8500.2 enclosure 4 at hand as Table 4-3 only has the IA control acronym. In the table, "(dev)" means development environment, and "(op)" means operational environment. IA controls marked with "(A1)" are from MAC level 1, included in 8500.2 as A1; those marked with "(A4)" are from the confidentiality controls for classified information, included in 8500.2 as A4. Those included in both are marked accordingly. An IA Control is included in the DIACAP table below if:

- It is an AIS system, or its infrastructure, per 8500.2.
- It directly provides artifacts, measures, or evidence to support the system assurance case.
- It includes both development and operational environments.
- It includes both defensive (security) and intrinsic (mission) functions.
- DIACAP certification is a DoD requirement, and thus mandates the inclusion of references to controls that are (1) the generic assurance requirements in 3.4 and (2) those DoD-unique assurance requirements.

Table C-2 Mapping of DIACAP to IA Controls and DoD Life Cycle

DIACAP Step	DIACAP Activities	DAG Life Cycle Phase	DAG Review	IA Controls
1. Initiate and Plan IA C&A	Register System with DoD Component IA Program	Technology Development	SRR	DCPB-1 (A1)
	Assign IA Control	Technology Development System Development & Demo	SRR / SRR	DCBP-1 (A1) DCFA-1 (A1) DCII-1 (A1)
	Assemble DIACAP Team	System Development & Demo	SRR / SRR	DCSD-1 (A1) PRTN-1 (A1)
	Initiate DIACAP Implementation Plan	System Development & Demo	PDR / CDR / TRR	DCID-1 (A1) DCII-1 (A1) DCNR-1 (A1)
2. Implement and Validate Assigned IA Controls	Execute DIACAP Implementation Plan	System Development & Demo	PDR / CDR / TRR / PRR	DCBP-1 (A1) DCCS-2 (dev) (A1) DCCT-1 (op) (A1) DCID-1 (A1) DCFA-1 (A1) DCMC-1 (A1) DCNR-1 (A1) DCPA-1 (A1) DCPD-1 (A1) DCPP-1 (A1) DCPR-1 (A1) DCSL-1 (A1)
	Conduct Validation Activities		CDR / TRR / PRR	
	Prepare POA&M		CDR / TRR / PRR	
	Compile Validation Results in DIACAP scorecard		TRR / PRR	

DIACAP Step	DIACAP Activities	DAG Life Cycle Phase	DAG Review	IA Controls
				DCSP-1 (A1) DCSQ-1 (A1) DCSS-2 (A1) IATS-2 (A1) ECCD-2 (A1 and A4) ECIM-1 (A1) ECPA-1 (A1) ECPC-2 (A1) ECRG-1 (A1) ECSD-2 (A1) ECTM-2 (A1) ECTP-1 (A1) ECVI -1 (A1) ECVP-1 (A1) ECWN-1 (A1) PESL-1 (A1) PRRB-1 (A1) COTR-1 (A1) VIVM-1 (A1) ECAT-2 (A1 and A4) DCII-1 (A1) DCAS-1 (A4) DCSR-3 (A4) IAGA-1 (A1) IAIA-2 (A1) IAKM-3 (A4) ECAN-1 (A4) ECAR-3 (A4) ECCM-1 (A4) ECCR-2 (A4) ECCT-2 (A4) ECIC-1 (A4) ECLC-1 (A4) ECLO-2 (A4) ECLP-1 (A4) ECML-1 (A4) ECMT-2 (A4) ECNK-1 (A4) ECNK-2 (A4) ECRC-1 (A4) ECTC-1 (A4) IAAC-1 (A4) EBRP-1 (A4) EBRU-1 (A4) PECF-2 (A4) PESS-1 (A4) PEVC-1 (A4) PRAS-2 (A4) PRNK-1 (A4)
3. Make	Make Certification	System	PRR / OTRR	DCCB-2 (A1)

DIACAP Step	DIACAP Activities	DAG Life Cycle Phase	DAG Review	IA Controls
Certification Determination & Accreditation Decision	Determination Issue Accreditation Decision	Development & Demo (for Interim Authority to Operate) Production & Deployment (for Authority to Operate)		ECSC-1 (A1) IAGA-1 (A1)
4. Maintain Authorization to Operate and Conduct Reviews	Maintain Situational Awareness Maintain IA Posture Conduct Reviews (Review of IA Controls must occur at least annually) Initiate Re-accreditation	Operations & Support	OTRR / ISR	DCAR-1 (A1) DCBP-1 (A1) DCCS-2 (op) (A1) DCCT-1 (op) (A1) DCID-1 (A1) DCFA-1 (A1) DCPR-1 (A1) DCSL-1 (A1) DCSP-1 (A1) DCSQ-1 (A1) DCSS-2 (A1) IATS-2 (A1) ECIM-1 (A1) ECPA-1 (A1) ECPC-2 (A1) ECRG-1 (A1) ECSC-1 (A1) ECSD-2 (A1) ECTP-1 (A1) ECVI -1 (A1) ECVP-1 (A1) ECWN-1 (A1) PESL-1 (A1) PRRB-1 (A1) COTR-1 (A1) VIVM-1 (A1) DCSR-3 (A4) DCSS-2 (A4) IAGA-1 (A1) IAIA-2 (A1) IAKM-3 (A4) ECAN-1 (A4) ECAR-3 (A4) ECAT-2 (A1 and A4) ECCD-2 (A1 and A4) ECIC-1 (A4) ECLO-2 (A4) ECLP-1 (A4) ECML-1 (A4) ECMT-2 (A4) ECNK-1 (A4) ECNK-2 (A4)

DIACAP Step	DIACAP Activities	DAG Life Cycle Phase	DAG Review	IA Controls
				ECRC-1 (A4) IAAC-1 (A4) EBRU-1 (A4) PECF-2 (A4) PESS-1 (A4) PRAS-2 (A4) PRNK-1 (A4)
5. Decommission	Retire System	Operations & Support	ISR	PECS-2 (A1) PEDD-1 (A1)

The following controls are missing specific reviews: DCCB, DCPR, ECAT, ECPC, ECAN, PESP, PESS.

C.2 40 U.S.C. MAPPINGS

Table C-1 maps Title 40.U.S.C. (formerly, the Clinger-Cohen Act) requirements to system assurance requirements, and describes what should be done to support system assurance.

Table C-3 Mapping from Title 40.U.S.C. (formerly, the Clinger-Cohen Act) Requirements to System Assurance Requirements

Requirements Related to the Title 40.U.S.C. (formerly, the Clinger-Cohen Act) of 1996 (reference (l))	System Assurance Requirements	Applicable Milestone	Applicable Program Documentation **
*** Make a determination that the acquisition supports core, priority functions of the Department	N/A	Milestone A	ICD approval
*** Establish outcome-based performance measures linked to strategic goals	TB D until system assurance measures mature	Milestone A & B	ICD, CDD, CPD and APB approval
*** Redesign the processes that the system supports to reduce costs, improve effectiveness and maximize the use of COTS technology	Ensure that system assurance is addressed as part of the reengineering process, particularly with regard to COTS (See Initial Technical Review)	Milestone A & B	Approval of the ICD, Concept of Operations, AoA, CDD, and CPD
* No Private Sector or Government source can better support the function	Ensure that system assurance is	Milestone A & B	Acquisition Strategy, AoA

Requirements Related to the Title 40.U.S.C. (formerly, the Clinger-Cohen Act) of 1996 (reference (l))	System Assurance Requirements	Applicable Milestone	Applicable Program Documentation **
	addressed as part of the reengineering process, particularly with regard to outsourcing (See Supplier Assurance, Alternative Systems Review)		
* An analysis of alternatives has been conducted	Ensure that system assurance is addressed as part of the analysis process (See Alternative Systems Review)	Milestone A	AoA
* An economic analysis has been conducted that includes a calculation of the return on investment; or for non-AIS programs, a Life-Cycle Cost Estimate (LCCE) has been conducted.	Ensure that system assurance is addressed as part of the ROI calculation. (See Alternative Systems Review)	For MAIS: Milestone A & B, & FRPDR (or their equivalent) For non-MAIS: Milestone B or the first Milestone that authorizes contract award	Program LCCE Program Economic Analysis for MAIS
There are clearly established measures and accountability for program progress	Establish system assurance measures where practicable. (See Systems Requirements Review)	Milestone B	Acquisition Strategy APB
The acquisition is consistent with the Global Information Grid policies and architecture, to	Follow guidance provided in	Milestone A, B & C	APB (Interoperability KPP) C4ISP (Information Exchange Requirements)

Requirements Related to the Title 40.U.S.C. (formerly, the Clinger-Cohen Act) of 1996 (reference (l))	System Assurance Requirements	Applicable Milestone	Applicable Program Documentation **
include relevant standards	this system assurance document.		
The program has an information assurance strategy that is consistent with DoD policies, standards, and architectures, to include relevant standards	See Mapping of DoD Controls (DoDI 8500.2 to System assurance guidebook Material	Milestone A, B, C, FRPDR or equivalent	Information Assurance Strategy
To the maximum extent practicable, (1) modular contracting has been used, and (2) the program is being implemented in phased, successive increments, each of which meets part of the mission need and delivers measurable benefit, independent of future increments	Ensure that security is maintained when components from different phases interact. (See Acquisition Strategy)	Milestone B or the first Milestone that authorizes contract award	Acquisition Strategy
The system being acquired is registered	N/A	Milestone B, Update as required	Registration Database

This page intentionally left blank.

Glossary

Following is a selected list of SA terms. Where possible, the authors have used standard definitions; in some cases, however, it was necessary to modify the standard definitions to more accurately convey the meaning within the SA context.

Abuse: Intentional or reckless misuse (i.e., use in an unintended way), alteration, disruption, or destruction. (*CNSSI 4009*)

Accountability: The security goal that generates the requirement for actions of an entity to be traced uniquely to that entity. This supports non-repudiation, deterrence, fault isolation, intrusion detection and prevention, and after-action recovery and legal action. (NIST, SP 800-27A)

Anomaly: Any condition in the software's operation or documentation that departs from the expected. This expectation can come from the software's documentation (e.g., requirements specification, design documentation, user documentation), its source code or other development artifact(s), or from someone's perception or experience with the software (e.g., an observation of how the software operated previously). (IEEE Std 1012-1986, IEEE Std 1044-1993)

Anti-Tampering (AT): The Systems Engineering (SE) activities intended to prevent and/or delay exploitation of critical technologies in U.S. systems. These activities involve the entire life cycle of systems acquisition including research, design, development, testing, implementation, and validation of anti-tamper measures. Properly employed, anti-tamper measures add longevity to a critical technology by deterring efforts to reverse-engineer, exploit, or develop countermeasures against a system or system component (aka element). (DAU)

Architecture: The fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution. (ANSI/IEEE Std 1471-2000, Recommended Practice for Architectural Description of Software-Intensive Systems) If a system is being developed to fit inside a larger enterprise architecture, the environment includes the enterprise architecture and the system's architecture includes the interfaces to that enterprise architecture.

Argument: A justification that a given claim (or sub-claim) is true or false.

Assurance: See system assurance.

Assurance Case: The set of *claims* of critical system assurance properties (requirements of the system), *arguments* that justify the claims (including assumptions and context), and *evidence* supporting the arguments.

Availability: The degree to which the services of a system or component (aka element) are operational and accessible when needed by their intended/authorized users. In the context of security, availability pertains to authorized services/actions only, and the need for availability generates the requirement that the system or component is able to resist or withstand attempts at unauthorized deletion or denial of service, regardless of whether those attempts are intentional or accidental. (*Goertzel at al. 2006, Avizienis* et al. 2004, *IEEE Std 610.12-1990, NIST SP 800-27-Rev.A*)

Blind Buy: An acquisition in which the identity of the acquirer and/or user(s) is intentionally concealed from the supplier.

Claim: The critical system requirements for assurance, including the maximum level of uncertainty permitted for it.

Component: One of the parts or elements that make up a system. A component may be hardware or software and may be divisible into smaller components. Note: With regard to

software, the terms module, component, and unit are often used interchangeably, or defined to be sub-elements of one another. (Goertzel et al. 2006) Used in this document as a synonym for "element."

Compromise: A violation of the security policy of the system, or an incident in which any of the security properties of the system are violated or its dependability is intentionally degraded. (*Goertzel et al. 2006, CNSSI 4009*)

Confidentiality: The property that sensitive information is not disclosed to unauthorized individuals, entities, or processes. (NIST, FIPS 140-2)

Countermeasure: An action, device, procedure, technique, or other measure that reduces the vulnerability of a component or system. (Goertzel et al. 2006, CNSSI 4009)

Critical Component: See critical element.

Critical Element: An element of a critical system whose unintentional or intentional subversion/failure (possibly in combination with others) could cause the loss of essential system functionality or mission failure.

Critical Program Information: Elements or components of an RDA 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 5200.39)

Criticality: A relative measure of the consequences of a failure mode and its frequency of occurrence. (DAU)

Critical System: A system determined to have such vital importance that its engineering, production, evaluation, sustainment, assurance, acquisition, and subsequent operation must be governed by focused system assurance activities.

Defense-in-Depth: A strategy that combines people, technology, and operations capabilities to establish protective barriers that span different layers and dimensions of a system in its operational environment in order to isolate that system from potential sources of attack. (Goertzel 2006, *CNSSI 4009*)

Denial of Service: An action or series of actions that (1) prevents access to a software system by its intended/authorized users; (2) causes the delay of its time-critical operations; or (3) prevents any part of the system from functioning. (*CNSSI 4009, NIST SP 800-27-Rev.A*)

Dependability: The degree to which the system is operable and capable of performing functionality or of delivering a service that can justifiably be relied upon (i.e., trusted) to be correct. To achieve dependability, the system must be able to avoid service failures that are more frequent or severe, or longer in duration, than is acceptable to users. Dependability may be viewed according to different, but complementary, properties (or "instances of dependability") required for a system to be considered dependable:

- Availability
- Integrity
- Reliability
- Safety
- Maintainability
- Security

Two other properties are closely related to dependability:

- Survivability
- Trustworthiness

For systems, availability and reliability emphasize the avoidance of failures, safety emphasizes the avoidance of a specific class of failures (catastrophic failures), and security emphasizes the prevention of exploitable development faults (intentional and unintentional) and malicious external faults. (Goertzel et al. 2006, *Avizienis* et al. 2004; *IFIP WG 10.4, TM 5-698-2*)

EIA 632, Processes for Engineering a System: Provides an integrated set of fundamental processes to aid a developer in engineering or re-engineering a system.

Element: See system element.

Enabling System: A system that complements a system-of-interest during its life cycle stages but does not necessarily contribute directly to its function during operation. (ISO/IEC 15288:2008)

Engineering-in-Depth (EiD): The employment of systems engineering processes to meet system assurance requirements, usually involving issues of incorporation of elements with different degrees of uncertainty about their relevant behavior or properties and defense-in-depth.

Enterprise: One or more organizations sharing a definite mission, goals, and objectives to offer an output such as a product or service (ISO 15704, Industrial automation systems — Requirements for enterprise-reference architectures and methodologies).

Enterprise Architecture (EA): The architecture is a conceptual blueprint that defines the structure and operation of an organization. The intent of an enterprise architecture is to determine how an organization can most effectively achieve its current and future objectives. EA is also variously defined as "…understanding all of the different elements that go to make up the enterprise and how those elements interrelate."[5]

Evidence: Information that demonstrably justifies the arguments in an assurance case.

IEEE 1220: Standard for managing systems engineering.

Information Assurance (IA): Measures that protect and defend information and information systems by ensuring their availability, integrity, authentication, confidentiality, and nonrepudiation. (NSTISSI 4009)

Integrity: The quality of a system or component that reflects its logical correctness and reliability, completeness, and consistency. In security terms, integrity generates the requirement for the system or component to be protected against either intentional or accidental attempts to (1) alter, modify, or destroy it in an improper or unauthorized manner, or (2) prevent it from performing its intended function(s) in an unimpaired manner, free from improper or unauthorized manipulation. (Goertzel et al. 2006, CNSSI 4009, NIST SP 800-27-Rev.A)

ISO/IEC 15288:2008, Systems and software engineering – System life cycle processes : Establish a common framework for describing the life cycle of systems.

Least Privilege: Principle requiring that each subject be granted the most restrictive set of privileges needed for the performance of that subject's authorized tasks. Application of this

[5] E.g., see: http://www.enterprise-architecture.info/EA_Methods.htm.

principle limits the damage that can result from accident, error, or unauthorized use of a component or system. (Goertzel et al. 2006, CNSSI 4009)

Maintainability: The ease with which a software system or component can be modified to correct faults, improve performance, or other attributes, or adapt to a changed environment (IEEE 90).

Malicious Code: Software or firmware intended to perform an unauthorized process that will have adverse impact on the dependability of a component or system. (Goertzel et al. 2006, *CNSSI 4009*)

Mission Assurance: Identify and mitigate design, production, test, and field support deficiencies that could affect mission success.

Non-Repudiation: In law, non-repudiation implies one's intention to fulfill his/her obligations to a contract. It also implies that one party of a transaction cannot deny having received a transaction nor can the other party deny having sent a transaction. Electronic commerce uses technology such as digital signatures and encryption to establish authenticity and non-repudiation.

Non-Developmental Items: The statutory definition of non-developmental item includes COTS and GOTS. See Federal Acquisition Regulations (Part 11) for further clarification.

Program Protection Plan (PPP): A risk-based, comprehensive, living plan to safeguard CPI that is associated with a RDA program. The PPP is used to develop tailored protection guidance for dissemination and implementation throughout the program for which it is created. The layering and integration of the selected protection requirements documented in a PPP provide for the integration and synchronization of CPI protection activities throughout the Department of Defense.

Reliability: The ability of a system and its parts to perform its mission without failure, degradation, or demand on the support system. (DAU)

Requirement: A statement that identifies a product or process operational, functional, or design characteristic or constraint, which is unambiguous, testable, or measurable, and necessary for product or process acceptability (by consumers or internal quality assurance guidelines). (IEEE 1220)

Risk: The possibility or likelihood that a particular threat will adversely impact a system by exploiting a particular vulnerability. (Goertzel et al. 2006, CNSSI 4009).

Risk: A measure of the inability to achieve program objectives within defined cost and schedule constraints. Risk is associated with all aspects of the program, e.g., threat, technology, design processes, or Work Breakdown Structure (WBS) elements. It has two components: the probability of failing to achieve a particular outcome, and the consequences of failing to achieve that outcome. (DAU)

Risk Management: All plans and actions taken to identify, assess, mitigate, and continuously track, control, and document program risks. (DAU)

Robustness: The degree to which a component or system can function correctly in the presence of invalid inputs or stressful environmental conditions, including inputs or conditions that are intentionally and maliciously created (IEEE Std 610.12-1990)

Sandboxing: A method of isolating application-level components into distinct execution domains, the separation of which is enforced by software. When run in a "sandbox," all of the "sandboxed" program's code and data accesses are confined to memory segments within that "sandbox." In this way, "sandboxes" provide a greater level of isolation between executing

processes than can be achieved when processes run in the same virtual address space. The most frequent use of sandboxing to isolate the execution of untrusted programs (e.g., mobile code, programs written in potentially unsafe languages such as C) so that each program is unable to directly access the same memory and disk segments used by other programs, including the application's trusted programs. Virtual machines (VMs) are sometimes used to implement sandboxing, with each VM providing an isolated execution domain. (Goertzel et al. 2006, NIST SP 800-19)

Secure State: The condition in which no subject can access another entity in an unauthorized manner. (Goertzel 2006, CNSSI 4009)

Security: Protection against intentional subversion or forced failure. Security is a composite of four attributes – confidentiality, integrity, availability, and accountability -- plus aspects of a fifth, usability, all of which have the related issue of their assurance. (Goertzel 2007, Redwine 2006)

Software: Software for these purposes shall be taken to mean the programs, routines, and symbolic languages that control the functioning of the hardware and direct its operation, including but not limited to the genre of items called software, firmware, microcode, source code, object code, machine code, machine language, etc. (from draft Army Software Safety Standard).

Software Assurance (SwA): (1) concerning uncertainty related to properties or behavior of software; (2) grounds for justified confidence in software's correctness of specification or meeting of specification or requirement.

Software Intensive: A software-intensive system is any system in which software contributes essential influences to the design, construction, deployment, and evolution of the system as a whole. Ref: IEEE STD 1471-2000.

Supply Chain: The set of organizations, people, activities, information, and resources for creating and moving a product or service, including its elements, from suppliers through to their customers.

System: A combination of interacting elements organized to achieve one or more stated purposes. (ISO/IEC 15288:2008)

System Assurance: The justified confidence that the system functions as intended and is free of exploitable vulnerabilities, either intentionally or unintentionally designed or inserted as part of the system at any time during the life cycle.

System Assurance Activities: A planned, systematic set of multi-disciplinary activities to achieve the acceptable measures of system assurance and manage the risk of exploitable vulnerabilities.

System Element: A member of a set of elements that constitutes a system. NOTE: A system element is a discrete part of a system that can be implemented to fulfill specified requirements. A system element can be hardware, software, data, humans, processes (e.g., processes for providing service to users), procedures (e.g., operator instructions), facilities, materials, and naturally occurring entities (e.g., water, organisms, minerals), or any combination. (ISO/IEC 15288:2008). Note that a system element can be composed of other system elements.

System-of-Interest: The system whose life cycle is under consideration in the context of this International Standard (ISO/IEC 15288:2008)

System of Systems (SoS) and **Family of Systems (FoS):** These terms are substantially defined elsewhere. For the purpose of this paper, SoS and FoS require additional diligence in system assurance because of the potential complexity involved. Aspects of SoS/FoS can sometimes

improve overall system assurance (e.g., multiple systems provided for recoverability in a FoS), while some aspects can also make assurance more difficult (e.g., complex interoperability among distributed SoS elements).

Threat Agent: Individual or organization that has an intent to cause harm

Trusted Foundry: The Trusted Foundry Program is a combined DoD-NSA project to develop and manufacture application-specific integrated circuits (ASICs) for critical DoD systems in a secure industrial environment. The Trusted Foundry process assures ASIC integrity from development and design through final delivery from NSA designated ASIC production facilities. (see FY2008 OSD RDT&E BUDGET ITEM JUSTIFICATION (R2 Exhibit)) http://www.dtic.mil/descriptivesum/Y2008/OSD/0605140D8Z.pdf)

Verification and Validation (V&V): The process of confirming, by examination and provision of objective evidence, that:

- Each step in the process of building or modifying the software yields the right products (verification). Verification asks and answers the question, "Was the software built right?" (i.e., correctness);
- The software being developed or modified will satisfy its particular requirements (functional and nonfunctional) for its specific intended use (validation). Validation asks and answers the question "Was the right software built?" (i.e., suitability).

(Goertzel et al. 2006, IEEE Std. 610.12-1990, ISO 9000, ISO/IEC 12207, NASA-GB-A201, NASA Glossary, NIST SP 500-234)

Vulnerability: A development fault or other weakness in a deployed system or exploited with malicious intent by a threat with the objective of subverting or incapacitating (violation of integrity, achieved by modification, corruption, destruction, or security policy violation)—often to gain access to the information it handles—or to force the software to fail (denial of service). Vulnerabilities can originate from flaws on the system's design, defects in its implementation, or problems in its operation. (Goertzel 2006, Avizienis et al, CIAO, CNSSI 4009, NIST SP 800-27-Rev. A, Sommerville, Veríssimo et al) (generalized to include systems, as well as software)

Weakness: A flaw, defect, or anomaly in a system that has the potential of being exploited as a vulnerability when the software is operational. A weakness may originate from a flaw in the system's security requirements or design, a defect in its implementation, or an inadequacy in its operational and security procedures and controls. The distinction between "weakness" and "vulnerability" originated with the MITRE Corporation Common Weaknesses and Exposures (CWE) project (http://cve.mitre.org/cwe/about/index.html). (Goertzel et al. 2006, Avizienis et al. 2004, CIAO, CNSSI 4009, NIST SP 800-27-Rev.A, Sommerville, Veríssimo et al. 2004) (generalized to include systems, as well as software).

Abbreviations and Acronyms

ACAT	Acquisition Category
ACE	Application Consulting and Engineering
ADM	Acquisition Decision Memorandum
ALARP	as low as reasonably practicable
AoA	Analysis of Alternatives
ASCAD	Adelard Safety Case Development
ASIC	application-specific integrated circuit
ASR	Alternative Systems Review
AT	anti-tamper
BIOS	basic input/output system
C&A	Certification and Accreditation
CAIG	Cost Analysis Improvement Group
CAPEC	Common Attack Pattern Enumeration and Classification
CCEVS	Common Criteria Evaluation and Validation Scheme
CDD	Capability Development Document
CDR	Critical Design Review
CI	configuration item
CI	counter intelligence
CIO	Chief Information Officer
CND	Computer Network Defense
COCOMO	Cost Control Model
CoI	Community of Interest
CoP	Community of Practice
COSECMO	Cost Security Model
COTS	commercial off-the-shelf
CM	configuration management
CMMI	Capability Maturity Model Integration
CTMM	Calculative Threat Modeling Methodology
CPI	critical program information
CR	Change Request
CVE	Common Vulnerabilities and Exposures
CWE	Common Weakness Enumeration
DAG	Defense Acquisition Guidebook
DIACAP	DoD Information Assurance Certification and Accreditation
DISA	Defense Information Systems Agency
DoD	Department of Defense
DoDD	Department of Defense Directive
DoDI	Department of Defense Instruction

DOTMLPF	Doctrine Organization Training Materiel Leadership Personnel and Facility
E2E	end to end
ECR	Engineering Change Request
EiD	Engineering in Depth
EM	Electromagnetic
EOIR	electro-optical and infrared
EPROM	Erasable PROM
FDCC	Federal Desktop Core Configuration
FIPS	Federal Information Processing Standards
FMEA	Failure Modes and Effects Analysis
FMECA	Failure Mode, Effects, and Criticality Analysis
FMICA	Failure Modes and Impacts Criticality Analysis
FoS	family of systems
FOUO	For Official Use Only
FPGA	field-programmable gate array
FRP	full-rate production
FTA	fault tree analysis
GIF	graphic interchange format
GIG	Global Information Grid
GOTS	government off-the-shelf
GSN	Goal Structuring Notation
HIPAA	Health Insurance Portability and Accountability Act of 1996
HW	Hardware
IA	information assurance
I&A	identification and authentication
IC	integrated circuit
IC	intelligence community
ICD	Initial Capabilities Document
ICSA	International Society for Computers and Their Applications
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IM	information management
IM	instant messaging
IMS	Integrated Master Schedule
INCOSE	International Counsel on Systems Engineering
INFOCON	Information Operations Condition
IP	Internet protocol
ISO/IEC	International Organization for Standardization / International Electrotechnical Commission

IT	information technology
ITAR	International Trafficking in Arms Regulation
ITR	Initial Technical Review
JCS	Joint Chiefs of Staff
KPP	Key Performance Parameter
LoA	Level of Assurance
LRIP	low-rate initial production
MOP	measures of performance
NDI	non-developmental item
NIAP	National Information Assurance Partnership
NIST	National Institute of Standards and Technology
NSA	National Security Agency
OS	operating system
OT&E	Operational Test & Evaluation
OTS	off-the-shelf
OWASP	Open Web Application Security Project
PDR	Preliminary Design Review
PLD	programmable logic device
PM	program manager; program management
PMBOK	Project Management Book of Knowledge
PPP	Program Protection Plan
PROM	Programmable ROM
RCM	Reliability-Centered Maintenance
R&D	requirements and development
RF	radio frequency
RFP	request for proposal
ROM	Read-Only Memory
SDP	Software Development Plan
SA	system assurance
SE	systems engineering; systems engineer
SEP	Systems Engineering Plan
SETR	Systems Engineering Technical Reviews
SFR	System Functional Review
SoS	system of systems
SOUP	system of unknown pedigree
SOX	Sarbanes-Oxley Act of 2002
SRR	System Requirements Reviews
SVR	System Verification Review
SW	Software
SwA	software assurance

SwACBK	Software Assurance Common Body of Knowledge
TDS	Technology Development Strategy
T&E	test and evaluation
TEMP	Test and Evaluation Master Plan
TES	Test and Evaluation Strategy
TRA	Technology Readiness Assessment
TRL	Technology Readiness Level
TR	Trouble Report
TRR	Test Readiness Review
VHDL	Very High Speed Integrated Circuit (VHDIC) Hardware Description Language
V&V	verification and validation
WBS	Work Breakdown Structure

References

Andrews, Richard A. Maintenance Planning (paper). Air Force Institute of Technology.

Avery, James, and Jim Holmes. 2006. *Windows Developer Power Tools.* O'Reilly.

Avizienis, Algirdas, Jean-Claude Laprie, Brian Randell, and Carl Landwehr. 2004. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing* 1(1):11–33.

Butler, R. W. 2006. *What Is Formal Methods?* http://shemesh.larc.nasa.gov/fm/fm-what.html.

CJCSI (Chairman of the Joint Chiefs of Staff Instruction) 6212.01D. 2006. Interoperability and Supportability of Information Technology and National Security Systems. March 8.

Colbert, Edward, and Danni Wu, 21st International Forum on COCOMO & Software Cost Modeling. Costing Secure Systems Workshop Report.

CNSSI 4009 (Committee on National Security Systems Instruction No. 4009). 2006. "National Information Assurance (IA Glossary." June. http://www.cnss.gov/Assets/pdf/cnssi_4009.pdf

Constantine, Larry L., and Scott W. Ambler. 2002. *The Unified Process Transition & Production Phases: Best Practices in Implementing the UP*. BMP Books.

CVE (Common Vulnerabilities and Exposures). http://cve.mitre.org/.

CWE (Common Weakness Enumeration). http://cwe.mitre.org/.

Defense Acquisition Guidebook. https://akss.dau.mil/dag/.

Defense Acquisition University (DAU). 2005. Glossary of Defense Acquisition Acronyms and Terms. 12th ed. Internet: http://www.dau.mil/pubs/glossary/12th_Glossary_2005.pdf or http://www.dau.mil/pubs/glossary/preface.asp

Deputy Secretary of Defense. 2003. "Defense Trusted Integrated Circuit Strategy." Memorandum, October 10.

DoD (Department of Defense) Dictionary of Military and Associated Terms. 2007. http://www.dtic.mil/doctrine/jel/doddict/

DoD 5200.1-M. Acquisition Systems Protection Program.

DoD 5220.22. 2006. National Industrial Security Program Operating Manual. February 28.

DoDD (Department of Defense Directive) 5000.1. 2003. The Defense Acquisition System. http://www.at.dod.mil/docs/d50001p.pdf.

DoDD 5105.21. System Threat Assessment.

DoDD 5200.39. Security, Intelligence and Counterintelligence Support to Acquisition Program Protection

DoDD 8500.01E Information Assurance (IA). http://www.dtic.mil/whs/directives/corres/pdf/850001p.pdf.

DoDI (Department of Defense Instruction) 5000.2. 2003. Operation of the Defense Acquisition System. https://akss.dau.mil/dag/DoD5002/Subject.asp.

DoDI 8500.2. Information Assurance (IA) Implementation. http://www.dtic.mil/whs/directives/corres/pdf/850002p.pdf

DoDI 8510.01. DoD Information Assurance Certification and Accreditation Process (DIACAP), Nov. 28, 2007.

DoD Information Assurance (IA) and Computer Network Defense (CND) Strategies: A Comprehensive Review of Common Needs and Capability Gaps–State-of-the-Art-Report (SOAR). July 21, 2005.

DSB (Defense Science Board). 2007. Mission Impact of Foreign Influence on DoD Systems.

EAWG (Financial Management Line of Business Taskforce Enterprise Architecture Working Group). 2004. Transition Approach Document, Office of Management and Budget. July 30.

EIA 632: http://www.techstreet.com/cgi-bin/detail?product_id=1145585

EIA 649: http://www.techstreet.com/cgi-bin/detail?product_id=1167865

Equipment Disposal Policy (draft). 2007. March 26. Gannon University.

Fedchak, Elaine, Thomas McGibbon, and Robert Vienneau. 2007. Software Project Management for Software Assurance. N.P.

FIPS (Federal Information Processing Standard) 200. 2005. Minimum Security Standard for Federal Information Systems. Gaithersburg, MD.: National Institute of Standards and Technology (NIST), U.S. Department of Commerce.

Goertzel, Karen Mercedes, et al. 2006. *Security in the software lifecycle: Making software development processes—and the software produced by them—more secure*. Draft 1.2 (August). Arlington, VA: U.S. Department of Homeland Security.

Hogganvik, Ida, and Ketil Stølen. 2006. *A Graphical Approach to Risk Identification, Motivated by Empirical Investigations. In 9th International Conference on Model Driven Engineering Languages and Systems* (MoDELS 2006). No. 4199 in *Lecture Notes in Computer Science*, pp. 574-588, Springer, 2006).

IEEE 1220. http://shop.ieee.org/ieeestore/Product.aspx?product_no=SS95334

IEEE P1633\AIAA R-013A. Standard for Software Reliability.

INCOSE (International Council on Systems Engineering). 2007. *Systems Engineering Handbook*. v.3. INCOSE.

INCOSE (International Council on Systems Engineering). 2005. *Guide to Systems Engineering Body of Knowledge* (G2SEBoK). http://g2sebok.incose.org/.

ISO/IEC (International Standards Organization/International Electrotechnical Commission) 9126. Parts 1, 2, and 3. Software Engineering: Product Quality

ISO/IEC 12207:2008. Systems and Software Engineering: Software Life Cycle Processes.

ISO/IEC 14598. Parts 1–6. Information Technology: Software Product Evaluation.

ISO/IEC 15288:2008. Systems and Software Engineering: System Life Cycle Processes.

ISO/IEC 15408–1:2005. Information Technology: Security Techniques. Part 1: Introduction and general model.

ISO/IEC 27001:2008. Information Security Management System Standard.

Luallen, Crit. 2005. State of Kentucky Auditors Alert on the Sanitation of Electronic Media. June.

Military Critical Technologies List. http://www.dtic.mil/mctl/.

MIL-HDBK-502. 1997 (May). Department of Defense Handbook for Acquisition Logistics.

NASA-STD-3000. 1995 (July). National Aeronautics and Space Administration, Man-Systems Integration Standards, Revision B.

NAVSO P-3690. 2001. Acquisition Logistics for the Rest of Us, an Assessment Tool from A to IOC. Office of the Assistant Secretary of Navy (RD&A). September.

NIST (National Institute of Standards and Technology) SP 800-53. 2005. *Recommended Security Controls for Federal Information Systems*. http://csrc.nist.gov/publications/PubsSPs.html

O'Brien, Frances, and Alvin R. Park. 2003. *Software Disposal: Old Software Never Dies.* Gartner.

OWASP (Open Web Application Security Project). 2005. *A Guide to Building Secure Web Applications*

and Web Services 2.0. The Open Web Application Security Project, Black Hat Edition. http://www.owasp.org/index.php/OWASP_Guide_Project

Pigoski, T.M.; Sexton, J. 1990. Software transition: A case study. In *Software Maintenance Proceedings* (Nov 26-29): 200–204.

PMBOK (*Project Management Book of Knowledge, A Guide to the*). 2004. 3d ed. Washington, DC: Project Management Institute.

Redwine, Samuel T., Jr., Ed. 2006. *Software Assurance: A Guide to the Common Body of Knowledge to Produce, Acquire, and Sustain Secure Software* (SwACBK). Version 1.1. U.S. Department of Homeland Security. https://buildsecurityin.us-cert.gov/daisy/bsi/resources/dhs/927.html

Rowland, L. Hawaii Pacific University.

SafSec-a. SafSec Methodology: Standard. (SafSec Standard). http://www.safsec.com/safsec_files/resources/50_2_SafSec_Method_Standard_3.0.pdf

SafSec-b. SafSec: Integration of Safety and Security. (SafSec Guidance). http://www.safsec.com/safsec_files/resources/50_3_SafSec_Method_ Guidance_Material_3.0.pdf

SANS Flash Announcement. March 20, 2007.

Swiderski, F., and W. Snyder. 2004. *Threat Modeling*. Microsoft Press.

Under Secretary of Defense (AT&L). 2004. "Encouraging Industry Participation in the Trusted Foundry Pilot Program." Memorandum, January 27.

Under Secretary of Defense (AT&L). 2004. "Expansion of Trusted Foundry Program." Memorandum, January 27.

Under Secretary of Defense (AT&L)/Assistant Secretary of Defense (NII). 2004. "Interim Guidance on Trusted Suppliers for Application Specific Integrated Circuits (ASICs)." Memorandum, January 27.

Walker, Ellen. 2005. DACS analyst, software development security: A risk management perspective. *DoD Software Tech News* (July). https://www.softwaretechnews.com/stn_view.php?stn_id=2&article_id=31

Other Resources

General

Abran, Alain, and James W. Moore, Eds. 2004. *Guide to the Software Engineering Body of Knowledge*. IEEE Computer Society. http://www.swebok.org

Anderson, James P. 1972. *Computer Security Technology Planning Study.* Vol. I, ESDTR-73-51, Vol. I, Bedford, MA: Electronic Systems Division, Air Force Systems Command, Hanscom Field, 01730.

Anderson, Ross J. 2001. *Security Engineering: A Guide to Building Dependable Distributed Systems*. John Wiley and Sons.

Ankrum, T.S., A.H. Kromholz, and Mitre Corp. 2005. Structured assurance cases: Three common standards. *Ninth IEEE International Symposium on High-Assurance Systems Engineering, 2005.* HASE, pp. 99- 108

Bell, David Elliot. 2005. Looking back at the Bell-La Padula Model. In *Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC '05)*, pp 337–351. http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel5/10467/33214/01565261.pdf

Bell, D. Elliott, and LaPadula, Leonard J. 1973. *Secure Computer Systems: Mathematical Foundations.* MITRE Corporation.

Bernstein, Lawrence, and C. M. Yuhas. 2005. *Trustworthy Systems Through Quantitative Software Engineering.* Wiley-IEEE Computer Society Press.

Bertino, Elisa, and Ravi Sandhu. 2005. Database security—Concepts, approaches, and challenges. IEEE

Transactions on Dependable and Secure Systems 2(1):2–19.

Biba, K. 1977. *Integrity Considerations for Secure Computer Systems*. ESD-TR-76-372, ESD/AFSC. Bedford, MA: Hanscom Air Force Base.

Bishop, Matt. 2003. *Computer Security: Art and Science.* Addison-Wesley.

Bishop, Matt, and Sophie Engle. 2006. The software assurance CBK and university curricula. In *Proceedings of the 10th Colloquium for Information Systems Security Education*.

Blanchard, Benjamin S. 2003. *Systems Engineering Management.* 3d ed. John Wiley and Sons.

Blanchard, Benjamin S., and Wolter J. Fabrycky. 2005. *Systems Engineering and Analysis*. Prentice Hall.

Buede, Dennis M. 2000. *The Engineering Design of Systems: Models and Methods.* New York: John Wiley and Sons.

Cannon, J. C. Privacy. 2005. *What Developers and IT Professionals Should Know.* Addison Wesley.

Clark, David D. and David R. Wilson. 1987. A comparison of commercial and military computer security policies. In *Proceedings of the 1987 IEEE Symposium on Security and Privacy*.

Cranor, Lorrie, and Simson Garfinkel. 2005. *Security and Usability: Designing Secure Systems That People Can Use.* O'Reilly.

Department of Defense Strategic Defense Initiative Organization (SDI). 1992. *Trusted Software Development Methodology.* SDI-S-SD-91-000007. Vol. 1 (17 June).

Department of Defense Trusted Computer System Evaluation Criteria (TCSEC). (Orange Book). http://csrc.nist.gov/publications/history/dod85.pdf

Director Central Intelligence Directive. 2000. Protecting Sensitive Compartmented Information within Information Systems (DCID 6/3).

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

Formal Methods. Formal Methods Publications.

Gasser, M. 1988. *Building a Secure Computer System.* Littleton, MA: Van Nostrand Reinhold.

Grance, Tim, Joan Hash, and Marc Stevens. 2004. *Security Considerations in the Information System Development Life Cycle*. Revision 1. NIST Special Pub 800-64. National Institute of Standards and Technology (NIST).

Gray, J. W. 1990. Probabilistic interference. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*. IEEE.

Guimaraes, Mario, Herb Mattord, and Richard Austin. 2004. Incorporating security components into database courses. In *Proceedings of the 1st Annual Conference on Information Security Curriculum Development.* ACM.

Hall, Anthony, and Rodrick Chapman. 2002. Correctness by construction: Developing a commercial secure system. *IEEE Software* 19(1):18–25.

Hoglund, Greg, and Gary McGraw. 2004. *Exploiting Software: How to Break Code.* Addison-Wesley.

Howard, Michael, and David C. LeBlanc. 2002. *Writing Secure Code.* 2d ed. Microsoft Press.

Howell, C. June 2005. Assurance Cases for Security. Follow-on workshop of the 2004 Symposium on Dependable Systems and Networks, June.

Ibrahim, Linda, et al. 2004. Safety and Security Extensions for Integrated Capability Maturity Models. Washington DC: U.S. Federal Aviation Administration.

IEEE (Institute of Electrical and Electronics Engineers). 1990. *IEEE Standard Computer Dictionary: A*

Compilation of IEEE Standard Computer Glossaries. New York, NY: IEEE.

ISO JTC 1/SC 27:2005. Information Technology—Security Techniques—A Framework for IT Security Assurance. Part 1: Overview and Framework. (ISO TR 15443 – 1). International Organization for Standardization (ISO).

ISO/IEC 9126:1991. Information Technology—Software Product Evaluation—Quality Characteristics and Guidelines for Their Use.

ISO/IEC 13888. Information Technology—Security Techniques—Non-Repudiation. (Parts 1–3).

Kossiakoff, Alexander, and William N. Sweet. 2003. *System Engineering Principles and Practice.* John Wiley and Sons.

Landwehr, Carl. 2001. Computer security. *IJIS* 1:3–13.

Lipner, Steve, and Michael Howard. 2005a. Microsoft Corporation, The Trustworthy Computing Security Development Lifecycle. Microsoft.

McGraw, Gary E. 2003. On the horizon: The DIMACS workshop on software security. *IEEE Security and Privacy.* March/April.

McLean, J. 1994. Security models. In *Encyclopedia of Software Engineering*, ed. J. Marciniak. Wiley.

Meier, J.D., et al. 2003. *Improving Web Application Security: Threats and Countermeasures.* Microsoft.

Merkow, Mark S., and Jim Breithaupt. 2005. *Computer Security Assurance Using the Common Criteria.* Thompson Delamr Learning.

Ministry of Defence. 2004. Interim Defence Standard 00-56, Safety Management Requirements for Defence Systems. Part 2: Guidance on Establishing a Means of Complying with Part 1, 17. December.

Ministry of Defence. 2003. Defence Standard 00-42 Issue 2, Reliability and Maintainability (R&M) Assurance Guidance. Part 3 R&M Case (6 June).

NASA (National Aeronautics and Space Administration). 1995. *Formal Methods Specification and Verification Guidebook for Software and Computer Systems*. Vol.1, *Planning and Technology Insertion.*

NASA. Software Assurance, Software Definitions. http://www.hq.nasa.gov/office/codeq/software/umbrella_defs.htm.

NASA. *Software Assurance Guidebook* (NASA-GB-A201). http://sato.gsfc.nasa.gov/guidebook/doc_view.php

NASA. *Software Assurance Guidebook and Standard* (NSA-std-2201-93). http://satc.gsfc.nasa.gov/assure/agb.txt & http://satc.gsfc.nasa.gov/assure/astd.txt

National Research Council (NRC) Computer Science and Telecommunications Board (CSTB). 2002. *Cybersecurity Today and Tomorrow: Pay Now or Pay Later*. National Academies Press.

National Security Agency (NSA). 2002. *The Information Systems Security Engineering Process.* (IATF) v3.1.

Naval Research Laboratory (NRL). 1995. *Handbook for the Computer Security Certification of Trusted Systems*. U.S. Naval Research Laboratory.

NCSC (National Computer Security Center). 2003. *A Guide to Understanding Covert Channel Analysis of Trusted Systems* (NCSC-TG-030, NCSC). NCSC.

NDIA (National Defense Industrial Association). 2005. "A White Paper on Software Assurance." Arlington, VA: NDIA. http://adm.omg.org/SoftwareAssurance.pdf

Neumann, Peter G. 1995. *Architectures and Formal Representations for Secure Systems.* Final Report, SRI Project 6401, October 2. SRI International Computer Sciences Laboratory.

NIST (National Institute of Standards and Technology) IR 7298. 2006. Glossary of Key Information Security Terms, ed. Richard Kissel. Internet (April 25, 2006): http://csrc.nist.gov/publications/nistir/NISTIR 7298_Glossary_Key_Infor_Security_Terms.pdf

NIST (National Institute of Standards and Technology) 800-26. 2001. *Security Self-Assessment Guide for Information Technology Systems*. http://csrc.nist.gov/publications/PubsSPs.html

NIST (National Institute of Standards and Technology) SP 800-33. 2001. *Underlying Technical Models for Information Technology Security*. http://csrc.nist.gov/publications/PubsSPs.html

NIST (National Institute of Standards and Technology) SP 800-23. 2000. *Guidelines to Federal Organizations on Security Assurance and Acquisition/Use of Tested/Evaluated Products.* Gaithersburg, MD: U.S. Department of Commerce. http://csrc.nist.gov/publications/PubsSPs.html

NIST (National Institute of Standards and Technology) SP 800-12. *CSD: SCRC, An Introduction to Computer Security - The NIST Handbook.* Chapter 9, Assurance. http://csrc.nist.gov/publications/nistpubs/800-12/800-12-html/chapter9.html

Older, Susan, and Shiu-Kai Chin. Building a Rigorous Foundation for Assurance into Information Assurance Education. White paper. Syracuse University Center for System Assurance.

Pfleeger, Charles P. and Shari Lawrence Pfleeger. 2003. *Security in Computing*. 3d ed. Prentice Hall.

Redwine, Samuel T., Jr. 2005. Creating a software assurance body of knowledge. *CrossTalk*. http://www.stsc.hill.af.mil/crosstalk/2005/10/0510Redwine.html

Redwine, Samuel T., Jr., and Noopur Davis (eds). 2004. *Processes for Producing Secure Software: Towards Secure Software.* Vols. 1 and 2. Washington, DC: National Cyber Security Partnership. http://www.nku.edu/~waldenj1/research/secure_software_engineering.html

Reed, Thomas C. 2004. *At the Abyss: An Insider's History of the Cold War*. Presidio.

Riguidel, Michel, et al. 2004. *D1.2 Assessment of Threats and Vulnerabilities in Networks.* Version 1.0. European Union Security Expert Initiative (SEINIT).

Ross, Ron, et al. 2004. *Guide for the Security Certification and Accreditation of Federal Information Systems.* NIST Special Publication 800-37. National Institute of Standards and Technology (NIST). http://csrc.nist.gov/publications/PubsSPs.html

Saltzer, J.H., and M.D. Schroeder. 1975. The protection of information in computer systems. In *Proceedings of the IEEE.* Vol. 63, no. 9, pp. 1278–1308. http://cap-lore.com/CapTheory/ProtInf/

Seminal papers: History of Computer Security Project, University of California Davis Computer Security Laboratory. http://seclab.cs.ucdavis.edu/projects/history/seminal.html

Sommerville, I. 2004. *Software Engineering*. 7th ed. Pearson Education.

SSE-CMM v.3.0, http://www.sse-cmm.org/docs/ssecmmv3final.pdf

Stoneburner, Gary, Clark Hayden, and Alexis Feringa. 2004. *Engineering Principles for Information Technology Security: A Baseline for Achieving Security.* Revision A. NIST SP 800-27.

Stroud, R., et al. 2004. A qualitative analysis of the intrusion tolerance capabilities of the MAFTIA architecture. *International Conference on Dependable Systems and Networks*. 453–461. http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel5/9172/29105/01311915.pdf

Thompson, Ken. 1984. Reflections on trusting trust. *CACM* 27:(8):761–763. http://www.ece.cmu.edu/~ganger/712.fall02/papers/p761-thompson.pdf

Vanfleet, W. Mark, et al. 2005. MILS: Architecture for high assurance embedded computing. *Crosstalk* (August). http://www.stsc.hill.af.mil/crossTalk/2005/08/0508Vanfleet_etal.html

Viega, John, and Gary McGraw. 2001. *Building Secure Software: How to Avoid Security Problems the Right Way.* Reading, MA: Addison Wesley.

Ware, Willis H. 1970. *Security Controls for Computer Systems (U): Report of Defense Science Board Task Force on Computer Security*. Santa Monica, CA: RAND Corporation.

Wasson, C.S. 2006. *System analysis, design, and development: Concepts, principles, and practices.* John Wiley and Sons.

Whittaker, J.A., and H.H. Thompson. 2003. *How to Break Software Security: Effective Techniques for Security Testing.* Addison Wesley.

Williams, Jeffrey R., and George F. Jelen. 1998. A framework for reasoning about assurance (23 April). National Security Agency. http://www.sse-cmm.org/docs/Framework.pdf

Information Security

Committee on National Security Systems (CNSS) Instruction (CNSSI) 4009. 2003. National Information Assurance (IA) Glossary. Revised (May).

ISO/IEC. 1998. System and Software Integrity Levels. ISO/IEC Standard 15026. http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=26236

ISO/IEC 12207:1995. Software Life Cycle Processes, Plus Amendment 1:2002 and Amendment 2:2004. http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=21208

Security Functionality

DoDI (Department of Defense Instruction) 8500.2. 2003. Information Assurance (IA) Implementation. Washington, DC: U.S. Department of Defense.

DOT&E (Director, Operational Test and Evaluation). 2006. "Policy for Operational Test and Evaluation of Information Assurance in Acquisition Programs" Memorandum, November 21.

FIPS 200 (Federal Information Processing Standards Publication 200). *Minimum Security Requirements for Federal Information and Information Systems.*

Supplier Assurance

DSB (Defense Science Board). 2007. Task Force on Mission Impact of Foreign Influence on DoD Software. *Mission Impact of Foreign Influence on DoD Software*. September.

DSB. 2005. "High Performance Microchip Supply." Defense Science Board (DSB) Task Force on High Performance Microchip Supply. February.

"Framework for Life Cycle Risk Mitigation for National Security Systems in the Era of Globalization." Center for National Security Systems (CNSS), Global Information Technology Working Group. November 2006.

GAO (Government Accountability Office). 2004. "Defense Acquisitions. Knowledge of Software Suppliers Needed to Manage Risks." May. GAO-04-678. Washington, DC: GAO. http://www.gao.gov/new.items/d04678.pdf

GAO (Government Accountability Office). 2004. "Defense Acquisitions. Stronger Management Practices are Needed to Improve DoD's Software-Intensive Weapons Acquisitions." March. GAO-04-393. Washington, DC: GAO. http://www.gao.gov/new.items/d04393.pdf

"Mandatory Procedures for Research and Technology Protection Within the DoD." DoD 5200.39-R.

"Protecting Sensitive Compartmented Information Within Information Systems." DCID 6/3, June 5, 1999.

U.S. President's Information Technology Advisory Committee. February 2005. "Cyber Security: A Crisis of Prioritization." Arlington, VA: National Coordination Office for Information Technology Research and Development. http://www.nitrd.gov/pitac/report/

Systems Engineering

Air Force Center for Systems Engineering: http://www.afit.edu/cse/

Alexander, Ian. 2001. Systems Engineering Isn't Just Software. http://easyweb.easynet.co.uk/~iany/consultancy/systems_engineering/se_isnt_just_sw.htm

Bahill, A.T., and B. Gissing. 1998. Re-evaluating systems engineering concepts using systems thinking. In *IEEE Transaction on Systems, Man and Cybernetics, Part C: Applications and Reviews.* Vol. 28(4), pp. 516–527.

Defense Acquisition University Systems Engineering Community of Practice: https://acc.dau.mil/CommunityBrowser.aspx?id=17608

Office of Management and Budget Federal Desktop Core Configuration Policy http://nvd.nist.gov/fdcc/index.cfm

Rechtin, E. 2000. *Systems Architecting of Organizations: Why Eagles Can't Swim.* Boca Raton, FL: CRC Press. http://portal.acm.org/citation.cfm?id=1239825&dl=&coll=

Web Sites

Build Security In. Department of Homeland Security (DHS). https://buildsecurityin.us-cert.gov/daisy/bsi/home.html.

CAPEC (Common Attack Pattern Enumeration and Classification). http://capec.mitre.org/.

Crosstalk. Journal of Defense Software Engineering. http://www.stsc.hill.af.mil/crosstalk/2007/07/index.html.
This journal often covers topics related to Software Assurance. Copies are available online. Several special issues of note are:

COTS Integration	June 2007
Requirements Engineering	Dec 2006
Software Acquisition	May 2007
Software Assurance	Sept 2006
Software Security	Mar 2007, Oct 2005

CVE, Common Vulnerabilities and Exposures (CVE®). http://cve.mitre.org/.
This site provides a list of the standard names for security vulnerabilities and exposures.

CWE, Common Weakness Enumeration. http://cwe.mitre.org/.
This site provides a dictionary of common software weaknesses. CVEs are instances of CWEs.

INCOSE (International Council on Systems Engineering). http://www.incose.org/

Making Security Measurable, http://makingsecuritymeasurable.mitre.org/.
This site provides pointers to many of the standardization efforts and other repositories.

NIST National Institute of Standards and Technology). http://www.csrc.nist.gov.

OMG Software Assurance, http://swa.omg.org/ This site contains slides from various OMG SwA meetings, including SwA Information Day.

OWASP (Open Web Application Security Project). http://www.owasp.org/index.php/Main_Page.
This site contains various resources, including papers, presentations, and the OWASP Top 10.

PSM (Practical Software Measurement). http://www.psmsc.com/.
This site has papers on software measurement and tools to facilitate software measurement.

STSC (Software Technology Support Center). http://www.stsc.hill.af.mil/.
This is the parent site to *CrossTalk* and SSTC (System & Software Technology Conference).

US-CERT (United States Computer Emergency Response Team). https://portal.us-cert.gov/.
This site contains slides for Software Assurance Working Groups and Software Assurance Forums.

Index

ENGINEERING FOR SYSTEM ASSURANCE

NDIA System Assurance Committee

National Defense Industrial Association
2111 Wilson Boulevard, Suite 400
Arlington, VA 22201