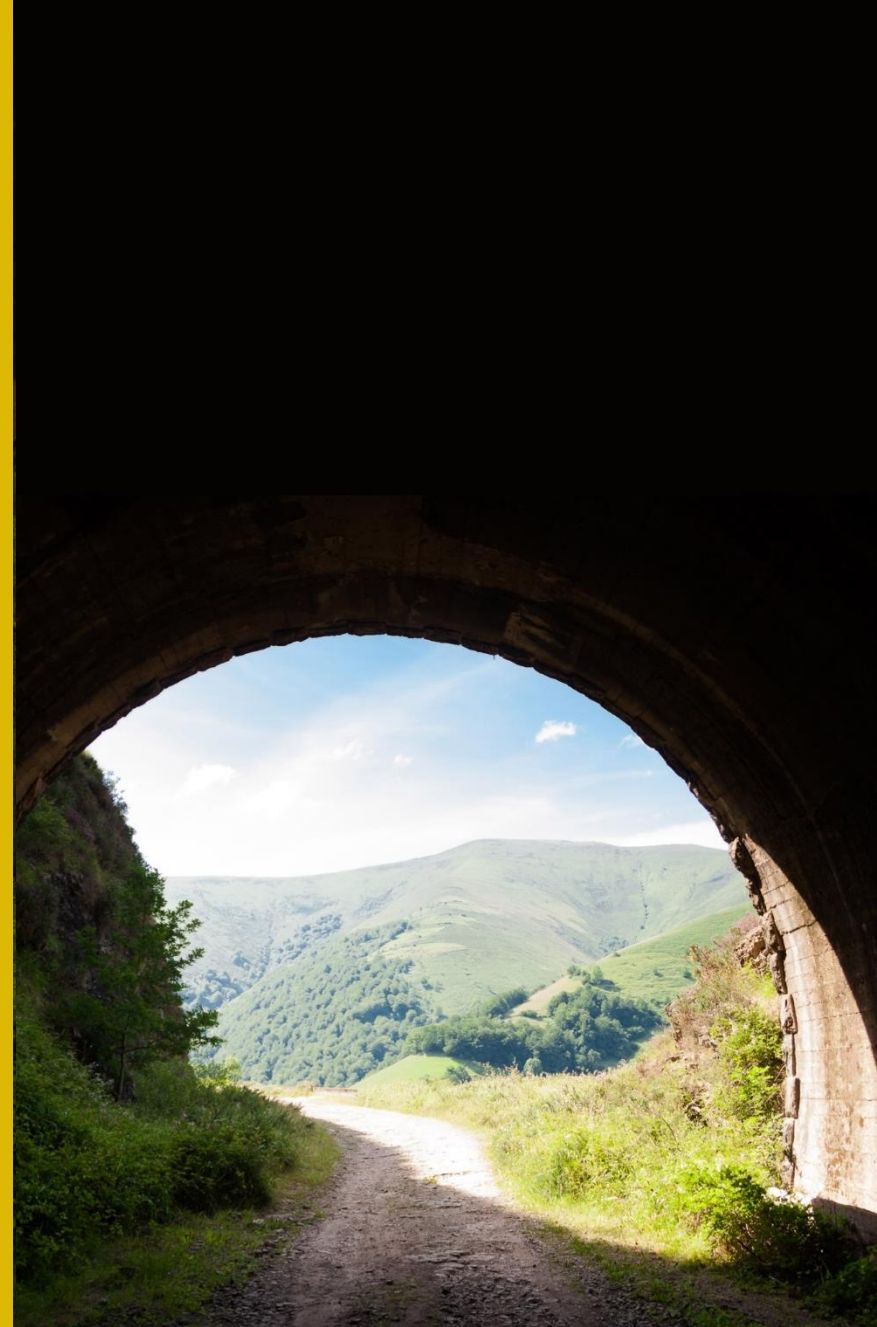


Software Assurance Methods in Support of Cybersecurity

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213



Software Engineering Institute

Carnegie Mellon University

© 2017 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution.

Copyright 2017 Carnegie Mellon University. All Rights Reserved.

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

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material is distributed by the Software Engineering Institute (SEI) only to course attendees for their own individual study.

Except for any U.S. government purposes described herein, this material SHALL NOT be reproduced or used in any other manner without requesting formal permission from the Software Engineering Institute at permission@sei.cmu.edu.

Although the rights granted by contract do not require course attendance to use this material for U.S. Government purposes, the SEI recommends attendance to ensure proper understanding.

Carnegie Mellon® and CERT® are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM17-0349



Software Engineering Institute

Carnegie Mellon University

© 2017 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution.

Software Assurance: Introduction





Part 1 – Introduction

Part 2 – SwA for Development

Part 3 – Mission Assurance

Part 4 – SwA for Acquisition

Part 5 - SwA Metrics

Course Summary

Department of Defense (DoD) definition:

“the level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at anytime during its lifecycle, and that the software functions in the intended manner.”

[CNSS Instruction No. 4009; DoDi 5200.44 p.12]

Software Assurance Curriculum Model definition:

Application of technologies and processes to achieve a required level of confidence that software systems and services function in the intended manner, are free from accidental or intentional vulnerabilities, provide security capabilities appropriate to the threat environment, and recover from intrusions and failures.

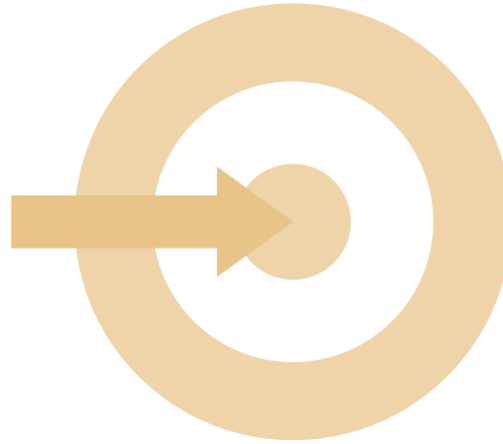
[Mead, Nancy; Allen, Julia; Ardis, Mark; Hilburn, Thomas; Kornecki, Andrew; Linger, Richard; & McDonald, James. *Software Assurance Curriculum Project Volume I: Master of Software Assurance Reference Curriculum*. CMU/SEI-2010-TR-005. Software Engineering Institute, Carnegie Mellon University. 2010. 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

Other variations:

- Software assurance is a level of confidence for
 - **Trustworthiness** - No exploitable vulnerabilities exist, either maliciously or unintentionally inserted.
 - **Predictable Execution** - When executed, software functions as intended.
- Software must function as intended and only as intended

What does SwA mean to you?

What is your involvement with SwA?



Develop awareness of the value for software assurance and the challenges of integrating it into the acquisition and development lifecycle

Develop an understanding of practices and methodologies available for addressing key areas of software assurance across the lifecycle

Increase awareness of ways to address critical lifecycle areas of software assurance

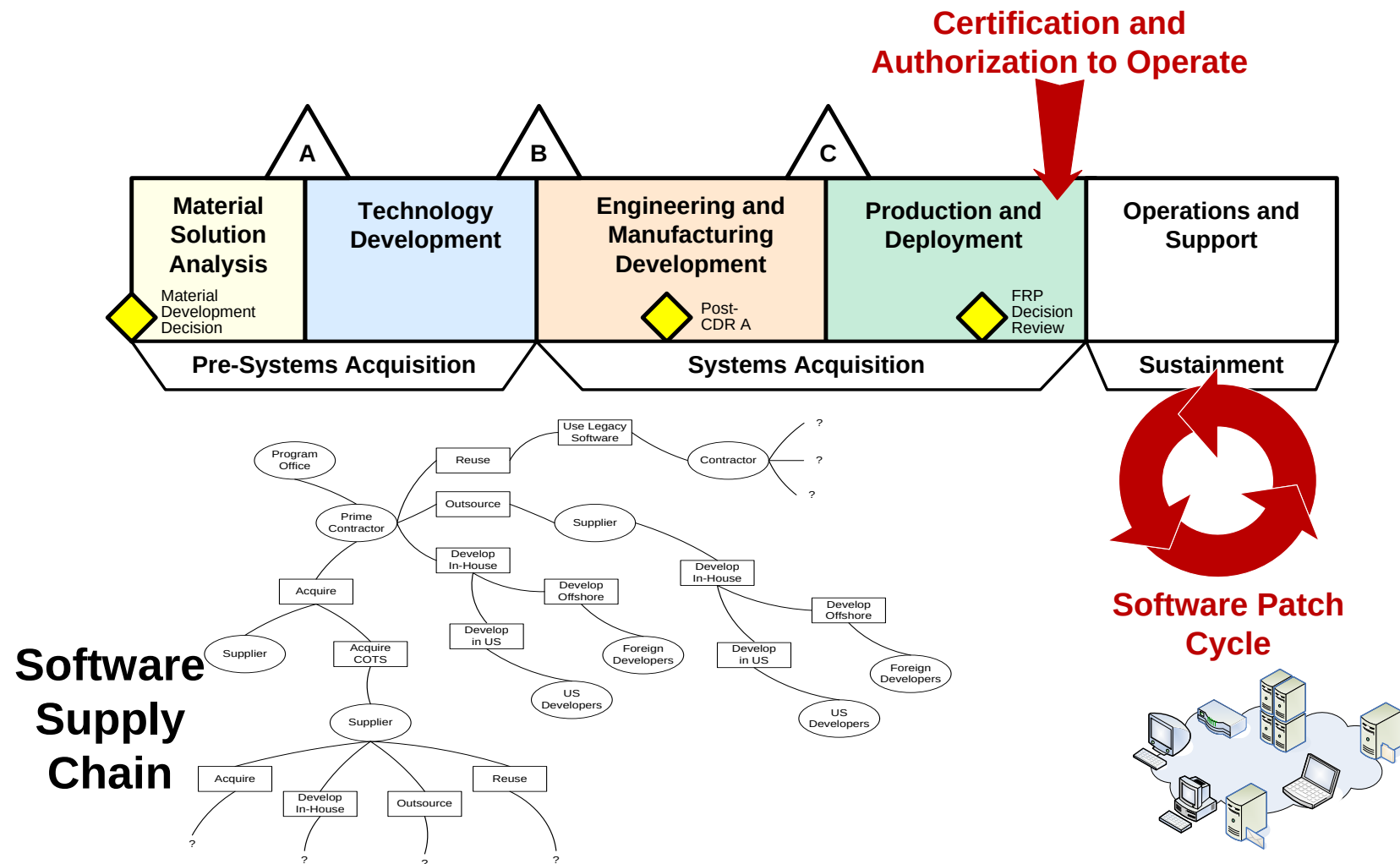
Recognize opportunities to leverage existing lifecycle capabilities to increase software assurance

Introduction

Software Assurance Landscape



Software Assurance Landscape: System Lifecycle



Each system is assumed to be self-sufficient

Reliability, quality, and effective systems engineering are sufficient for system development

Security requirements are selected:

- Based on system concerns for confidentiality, integrity, and availability (CIA) or mandated compliance
- Assigned to components through system engineering decomposition
- Software may be a sub-component of each component

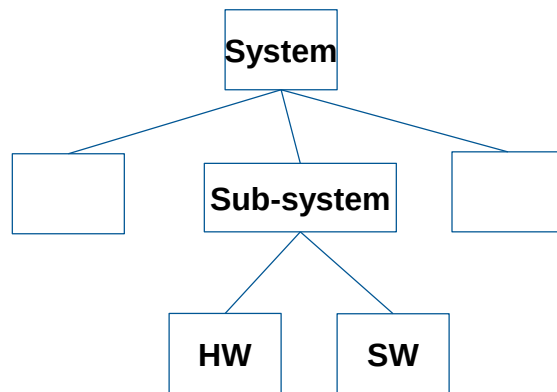
Components are assumed to be independent with well controlled interfaces

INCOSE. 2015. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 4.0. Hoboken, NJ, USA: John Wiley and Sons, Inc, ISBN: 978-1-118-99940-0

Software Engineering Does Not Enforce the Same Approach

Systems Engineering Assumptions

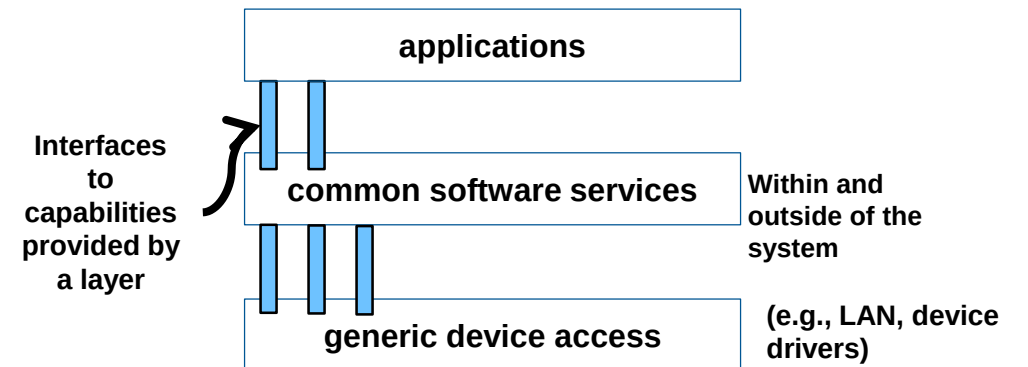
- Systems can be decomposed into discrete, independent, and hierarchically related components (or subsystems)
- **Is part of:** Components can be constructed and integrated with minimal effort based on the original decomposition
- **Quality properties can be allocated to specific components**



System Engineering Handbook
<http://www.incose.org/ProductsPublications/sehandbook>

Software Engineering Realities

- Software components are often related sets of layered functionality (one layer is *not* contained inside another layer)
- **Is used by:** Interactions of the components (*not* the decomposition) must be managed
- **Security properties relate to composite interactions (*not* to individual components)**



From the *NRC Critical Code Report* *

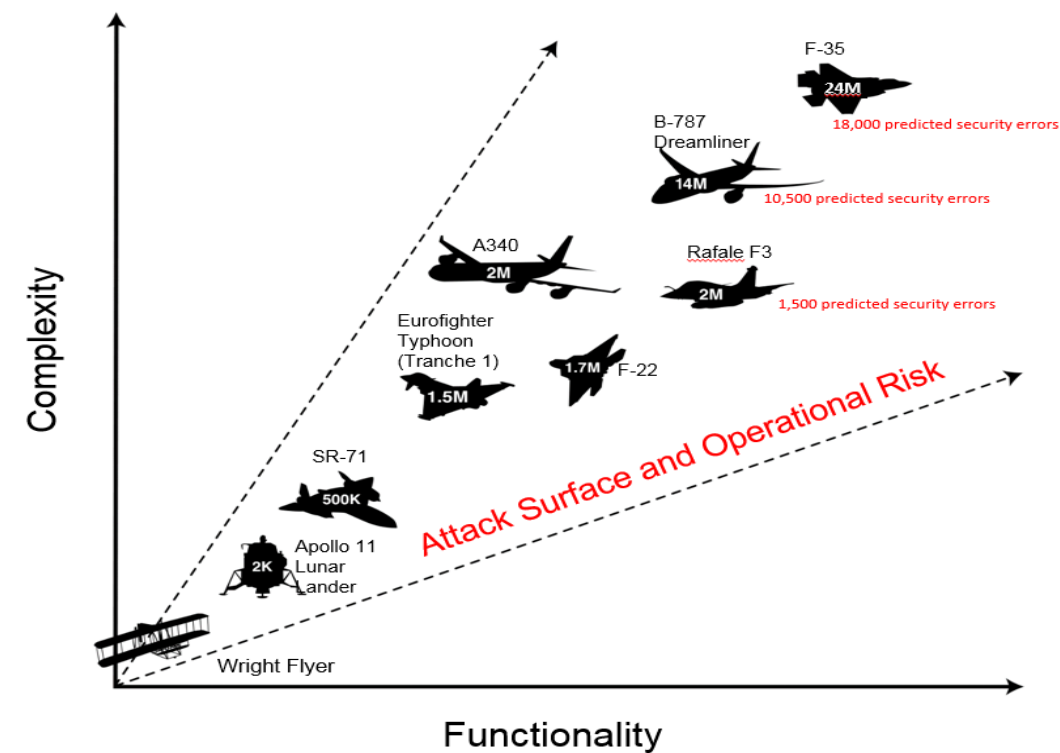
“Software has become essential to all aspects of military system capabilities and operations” p.19

1960 – 8% of the F-4 aircraft functionality

1982 – 45% of the F16 aircraft functionality

2000 – 80% of the F-22 aircraft functionality

* Committee for Advancing Software-Intensive Systems Producibility; National Research Council (NRC).
Critical Code: Software Producibility for Defense, 2010



Principles of security were defined by Saltzer and Schroeder in their paper “The Protection of Information in Computer Systems,” Communications of the ACM, **1974**

Security is defined as

“techniques that control who may use or modify the computer or the information contained in it”

Three main categories of concern:

confidentiality, integrity, and availability (CIA)

In 1974:

- S360 in use from 1964-1978; S370 came on market in 1972
- COBOL & BAL programming languages
- MVS operating system released in March 1974
- Patches were carefully tested to minimize operational disruption

Changes since 1974:

- Internet; Morris worm – November 2, 1988
- 50,000+ software vulnerabilities and exposures (CVE)
- Java, C++, C#
- Mobile and Cloud computing
- Patches are applied ASAP to minimize zero-day attacks

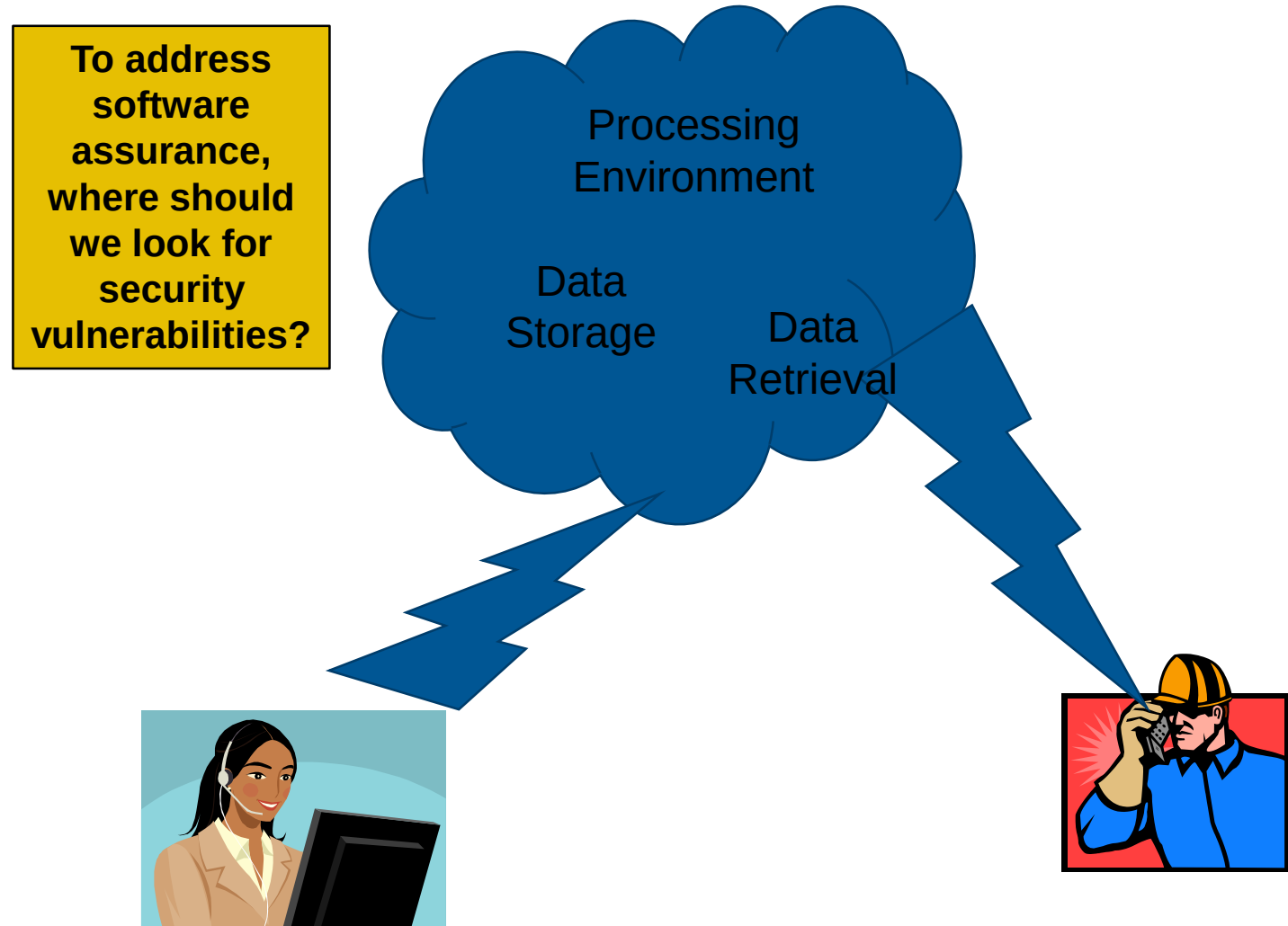
Definition: a weakness which allows an attacker* to bypass security controls

Requires three elements:

- a system susceptibility or flaw,
- attacker access to the flaw, and
- attacker capability to exploit the flaw

* Security Hacker. *Wikipedia*. July 2017. https://en.wikipedia.org/wiki/Security_hacker

What Is the System? Where Is the Software?



Individual component security is necessary but not sufficient
(*Development Assurance*)

Missions extend beyond a single system to define the
functionality that is intended and the operational impact of
software failure (*Mission Assurance*)

Software design and development is increasingly in the
supply chain, and supply chain risk management (SCRM)
must be part of the acquisition (*Acquisition Assurance*)

Introduction

Current Software Realities



There is no such thing as perfect code

- Best in class code contains 2.5 errors per function point or approximately 600 defects/MLOC
- Average quality is 4.5 errors per function point or approximately 6000 defects/MLOC

(Based on research by Capers Jones,
sqgnet.org/presentations/2011-12/Jones-Sep-2011.pdf)

There are too many possible combinations for code to be fully tested to confirm “functions as intended”

The threat landscape is constantly changing

- Evolving nature of threats, usage, and product functionality
- Skilled attackers know system weaknesses better than defenders

Under **ALL** operational circumstances?

Consider the following:

- System behavior outside of normal operating conditions, particularly for ad hoc and adverse conditions
- Accounting for constant change in the people/technology/software – mission outcome depends on continuous smooth interactions
- Sophisticated attacks that can exploit seemingly insignificant development and operational errors
- Sustainment and the changing threat landscape

Introduction

Software Assurance Principles



Risk
Interactions
Trusted Dependencies
Attacker
Coordination and Education
Well Planned and Dynamic
Measurable

Reference:

“Principles and Measurement Models for Software Assurance,” Nancy R. Mead, Dan Shoemaker (University of Detroit Mercy), Carol Woody,

2013 IJSSE Special Issue on Cybersecurity Scientific Validation, January 2013,
<http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=298843>



The perception of risk drives assurance decisions

- Assurance implementation choices (policies, practices, tools, restrictions) are based on the perception of threat and the impact should that threat be realized
- Perceptions are based on successful attacks
 - the current state of assurance is largely reactive
 - more successful organizations react and recover faster, learn from the reactive responses or others, and are more vigilant in anticipating and detecting attacks
- Misperceptions are failures to recognize threats and impacts – “how could it happen to us?” or “it could not happen here!”

Highly connected systems require alignment of risk across all stakeholders and systems otherwise critical threats will be unaddressed (missed, ignored) at different points in the interactions

- There are costs to addressing assurance which must be balanced against the impact of the risk
- Risk must also be balanced with other opportunities/needs (performance, reliability, usability, etc.)
- Interactions occur at many technology levels (network, security appliances, architecture, applications, data storage, etc.) and are supported by a wide range of roles

Your assurance depends on other peoples' assurance decisions and the level of trust you place on these dependencies:

- Each dependency represents a risk
- Dependency decisions should be based on a realistic assessment of the threats, impacts, and opportunities represented by an interaction
- Dependencies are not static and trust relationships should be reviewed to identify changes that warrant reconsideration
- Using many standardized pieces to build technology applications and infrastructure increases the dependency on others' assurance decisions



There are **no prefect protections against attacks**. There exists a broad community of attackers with growing technology capabilities able to compromise the confidentiality, integrity, and availability of any and all of your technology assets and the attacker profile is constantly changing.

- The attacker uses technology, processes, standards, and practices to craft a compromise (socio-technical responses)
- Attacks are crafted to take advantage of the ways we normally use technology or designed to contrive exceptional situations where defenses are circumvented

Assurance requires **knowledge of what can go wrong and effective risk coordination among all technology participants** and their governing bodies

- Protection must be applied broadly across the people, processes, and technology because the attacker will take advantage of all possible entry points
- Authority and responsibility must be clearly established at an appropriate level in the organization to ensure effective participation and coverage
- All participants must have appropriate competencies for software assurance



The capabilities to respond to a threat must be designed into the system and the threat is always changing.

Assurance implementation must represent a balance among governance, construction, and operation and is highly sensitive to changes in each of these areas

- Engineering challenge: Assurance cannot be added later; you must plan and build to the level of acceptable assurance that you need
- Continuous monitoring must be part of the planned response
- No one has resources to redesign systems every time the threat changes

A means to measure and monitor assurance must be built in.

If you can't measure it you can't manage it.

- All elements of the socio-technical environment must tie together (practices, processes, procedures, products, etc.) and measurements must be consistent
- Effective measurement is well supported by sound engineering and organizational principles – well-formed and consistently applied processes are critical to ensure an appropriate measurable response
- Measurement must be multi-faceted

Software Assurance: SwA for Development





Building Security In

Cybersecurity Lifecycle

Cybersecurity Extends to the Organization

Requirements Engineering Processes

Assured Design

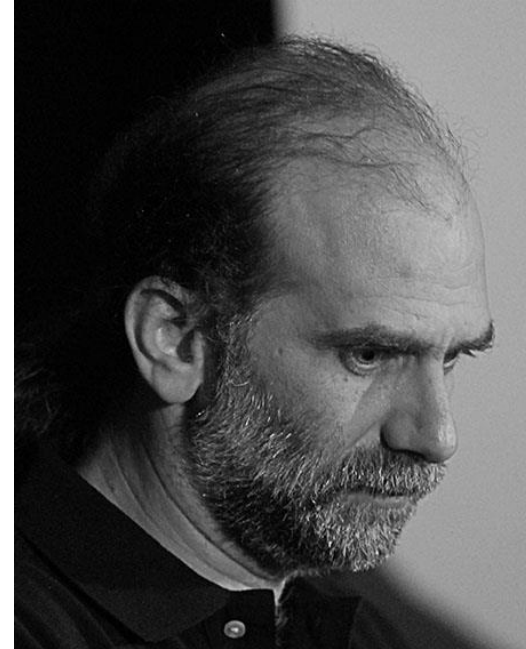
Code Analysis

Applying Software Quality Models

Summary

SwA for Development Building Security In





“We wouldn't have to spend so much time, money, and effort on network security if we didn't have such bad software security.”

**Bruce Schneier in Viega and McGraw,
“Building Secure Software,” 2001**

Source: Washington Post, March 19, 2014, http://www.washingtonpost.com/business/economy/toyota-reaches-12-billion-settlement-to-end-criminal-probe/2014/03/19/5738a3c4-af69-11e3-9627-c65021d6d572_story.html; <http://www.greene-broillet.com/Articles/Toyotasuddenacceleration.shtml>

DHS Build Security In (BSI)

<https://www.us-cert.gov/bsi>

Articles describing various aspects of sound SwA practices

- has been in existence since October 2005
- migrated from SEI to DHS supported platform Spring 2013
- review underway to remove obsolete materials and plan for enhancements



CERT Research Areas

- **Cybersecurity Engineering**
<http://www.cert.org/cybersecurity-engineering/>
- Secure coding <http://www.cert.org/secure-coding/>
- Insider threat <http://www.cert.org/insider-threat/>

The Building Security In Maturity Model (BSIMM) is a multi-year study (eight years and counting) led by Cigital and Fortify (<http://bsimm.com/>)

- Reviewed the efforts of 95 organizations from six verticals: financial services, independent software vendors, cloud, healthcare, Internet of Things, and insurance.
- Latest version, BSIMM-7, was released in October 2016.
- Objective was to identify what is currently done (state of the practice) rather than to promote specific [unproven] practices.
- Mechanism to compare yourself to industry peers for 113 practices

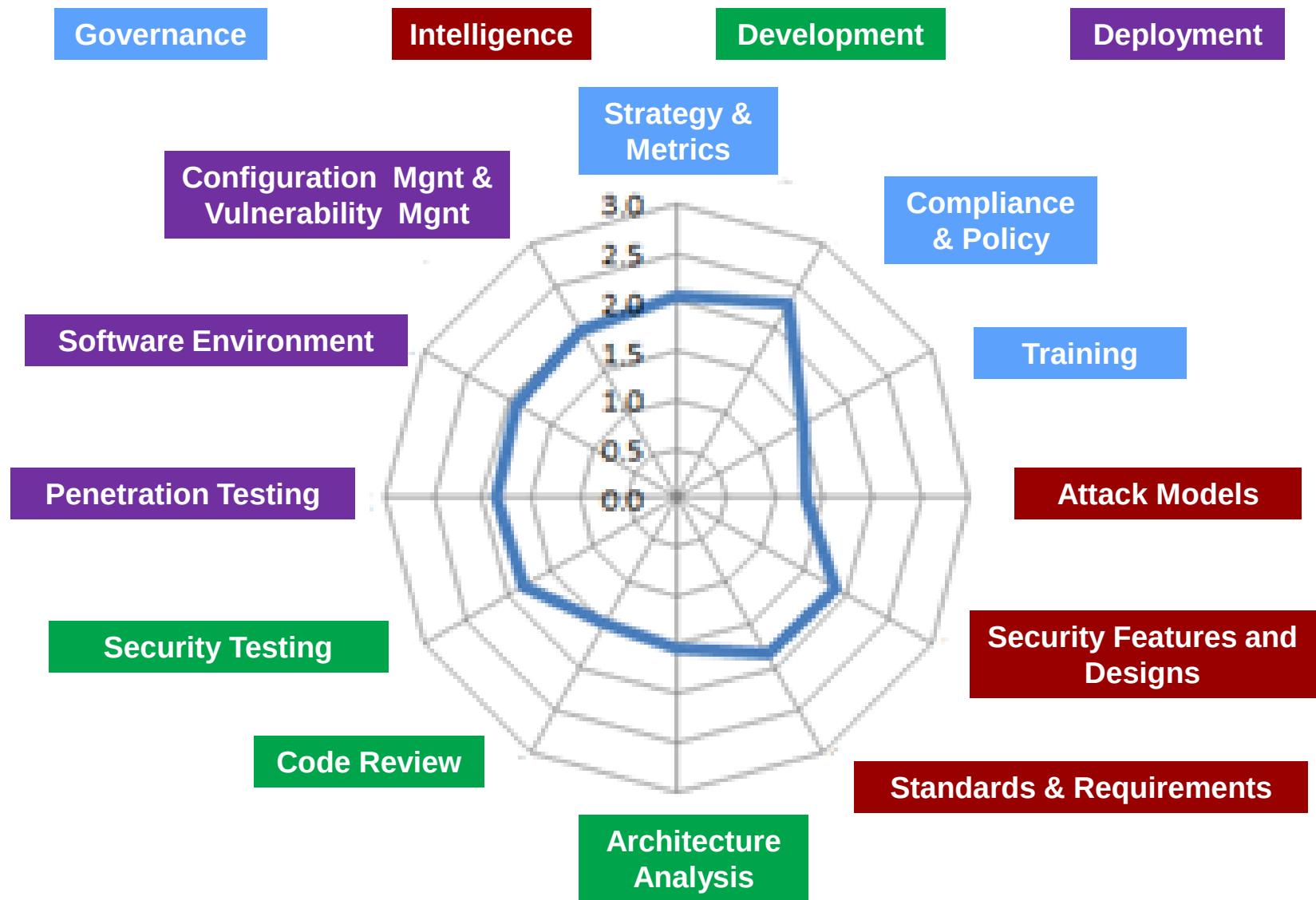
Identified 113 security activities across the surveyed organizations and organized those activities into 12 practice areas.

The number of collected activities demonstrates that there are multiple ways to improve software security and that the BSIMM activities should be used as specific acquisition requirements.

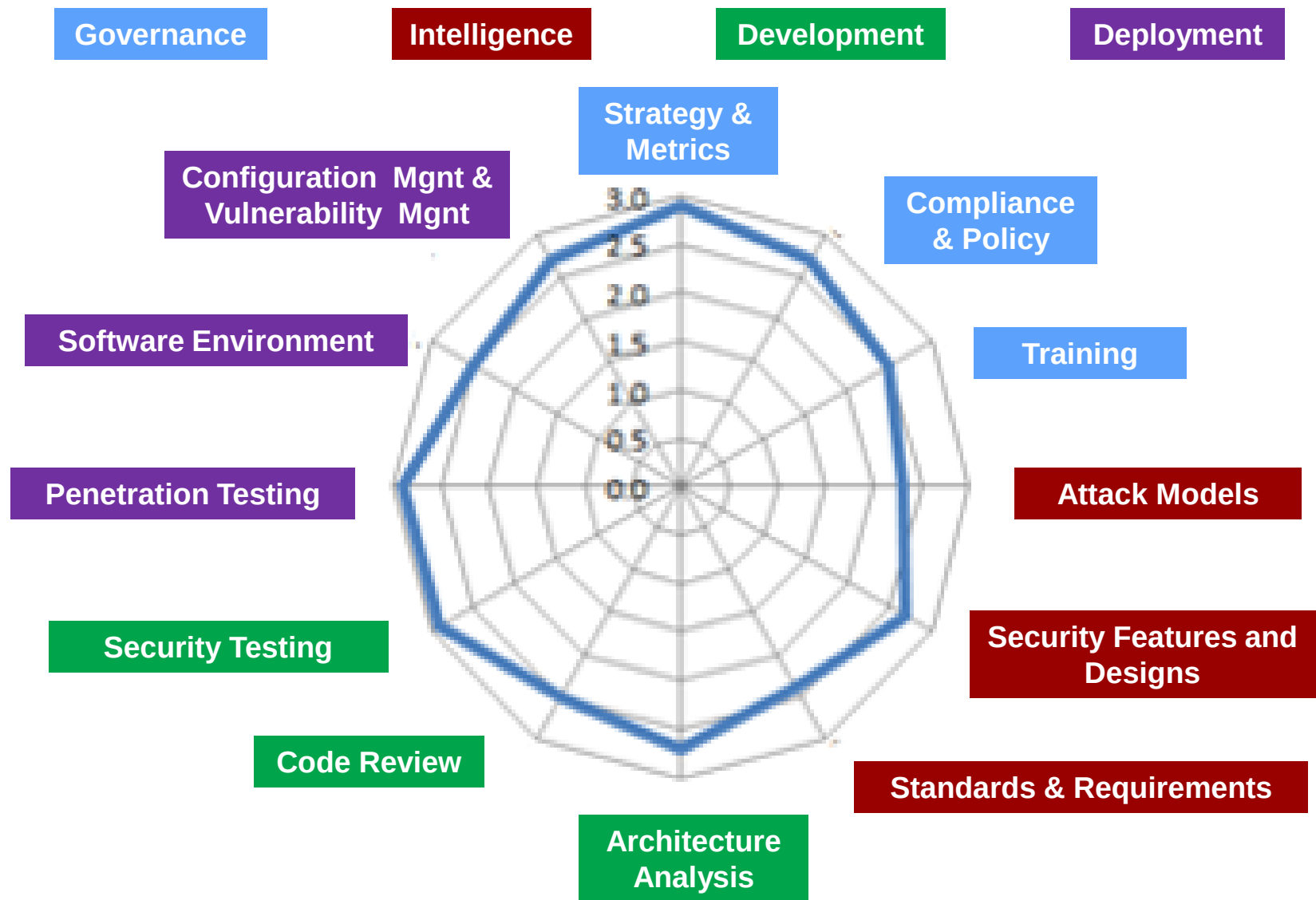
Defines three levels of levels of maturity for each practice based on usage.

- Level 1 activities (straightforward and simple) are commonly observed,
- Level 2 (more difficult and requiring more coordination) slightly less so
- Level 3 (rocket science) are much more rarely observed.

BSIMM Average Scores



BSIMM Scores for Top 10 Firms





**Object Management
Group**
(<http://www.omg.org/>)

**Software Assurance
Working Group** is
developing
specifications that
enable creation of
tools related to data
collection for
automation of
assurance



**Trusted Technology
Forum**
(<http://www.opengroup.org/subjectareas/trusted-technology>)

**Standard to certify
conformance to best
practices of ICT
Providers to mitigate
the risk of tainted &
counterfeit products.**



**SafeCode Industry
Consortium**
(<https://www.safecode.org/>)

**Major vendors (e.g. Microsoft,
Dell, Seimans, Adobe,
Symantec) who build critical
technology products share
information about how they
address security.**



NIST Software Assurance Metrics and Tool Evaluation (SAMATE)

(http://samate.nist.gov/index.php/Main_Page)

Sponsors Static Analysis Tool Exposition (SATE) to evaluate available tools against a standard set of vulnerabilities (Juliet suite) to encourage improvement of tools and speed adoption of tools by objectively demonstrating their use on real software



Software Assurance Marketplace

(<https://continuousassurance.org/>)

Developed to make it easier to consistently test the quality and security providing free ready-to-use cloud computing platform at mir-swamp.org or the SWAMP-in-a-Box (SiB) open-source distribution that is downloadable from GitHub

INSTITUTE FOR DEFENSE ANALYSES (IDA)

State-of-the-Art Resources (SOAR) for Software Vulnerability Detection, Test, and Evaluation

(<http://www.acq.osd.mil/se/docs/P-8005-SOAR-2016.pdf>)

Assist DoD and other government agencies in making effective software assurance (SwA) and software supply chain risk management (SCRM) decisions

ISC/ESC – secure coding, supply chain

NIST – security risk management, secure practices, security systems engineering, supply chain risk management

IEEE – software engineering

NDIA – system engineering

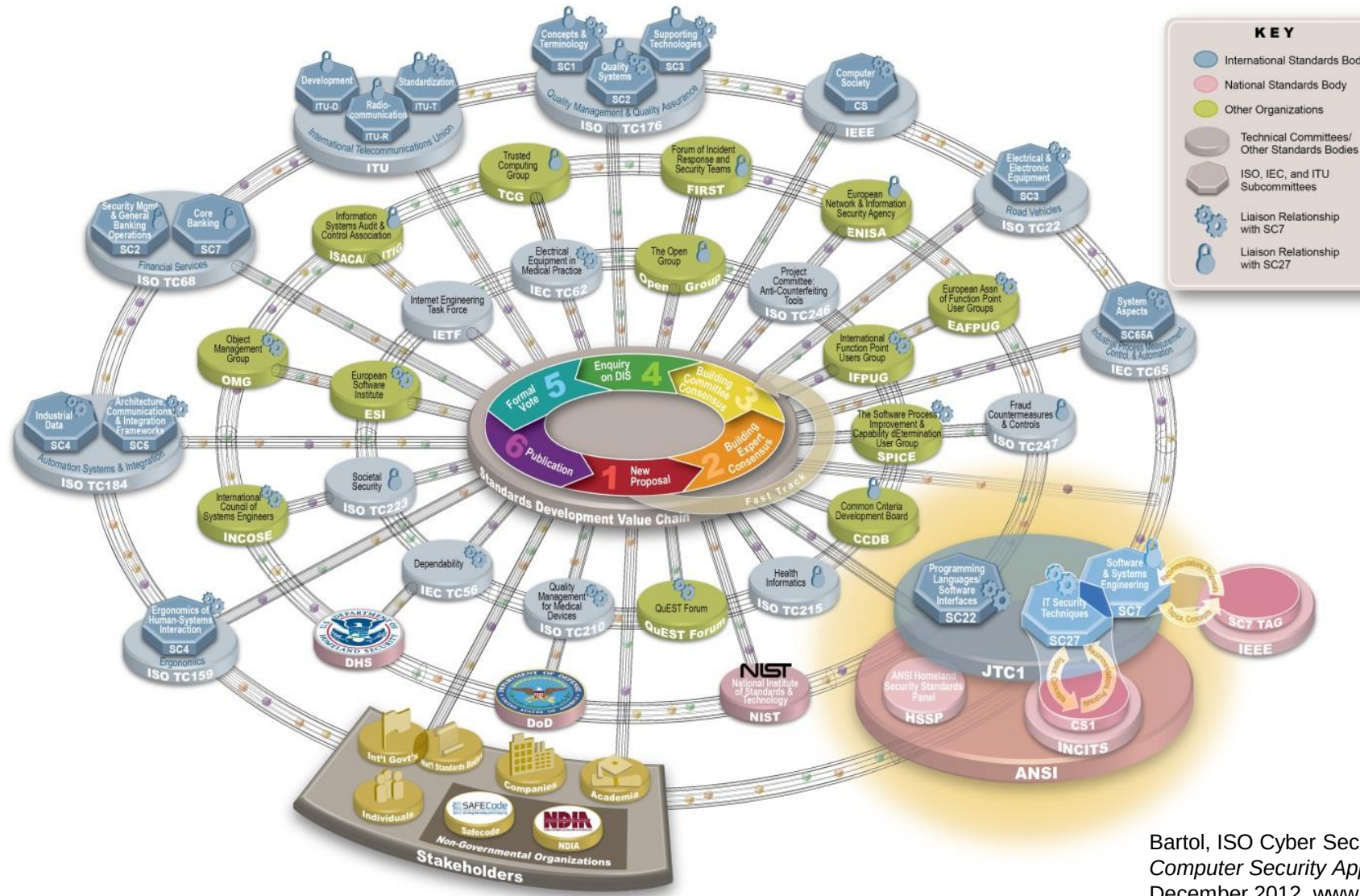
ISACA – process management (recently acquired CMMI)

ISC² – certification for information assurance, compliance with SwA curriculum model

INCOSE – system engineering

OWASP - Open Web Application Security Project

Standards Bodies

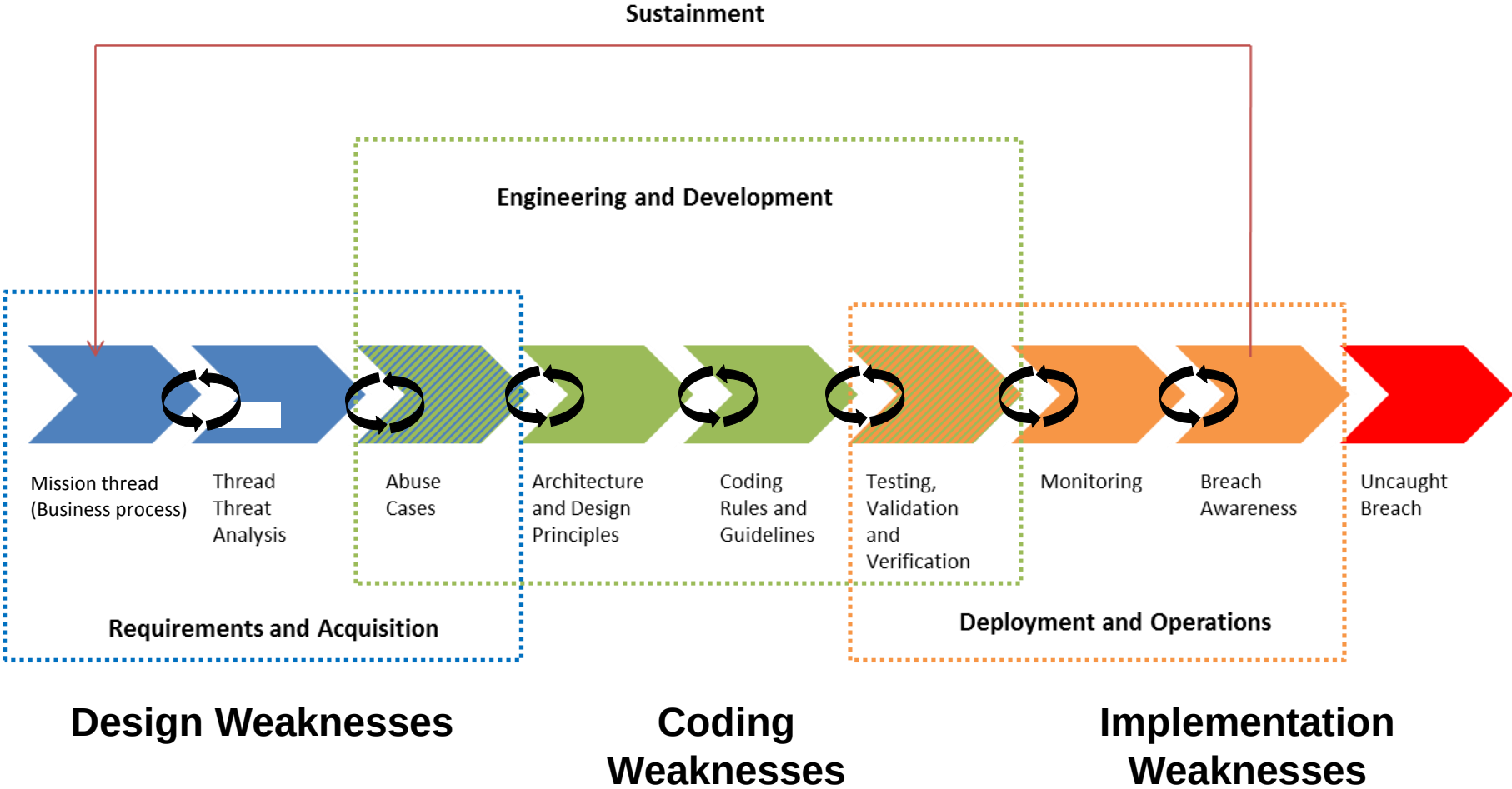


Bartol, ISO Cyber Security and ICT SCRM Standards, *Annual Computer Security Applications Conference (ACSAC)*, December 2012 www.acsac.org/2010/program/case/wed-1330-Bartol.pdf

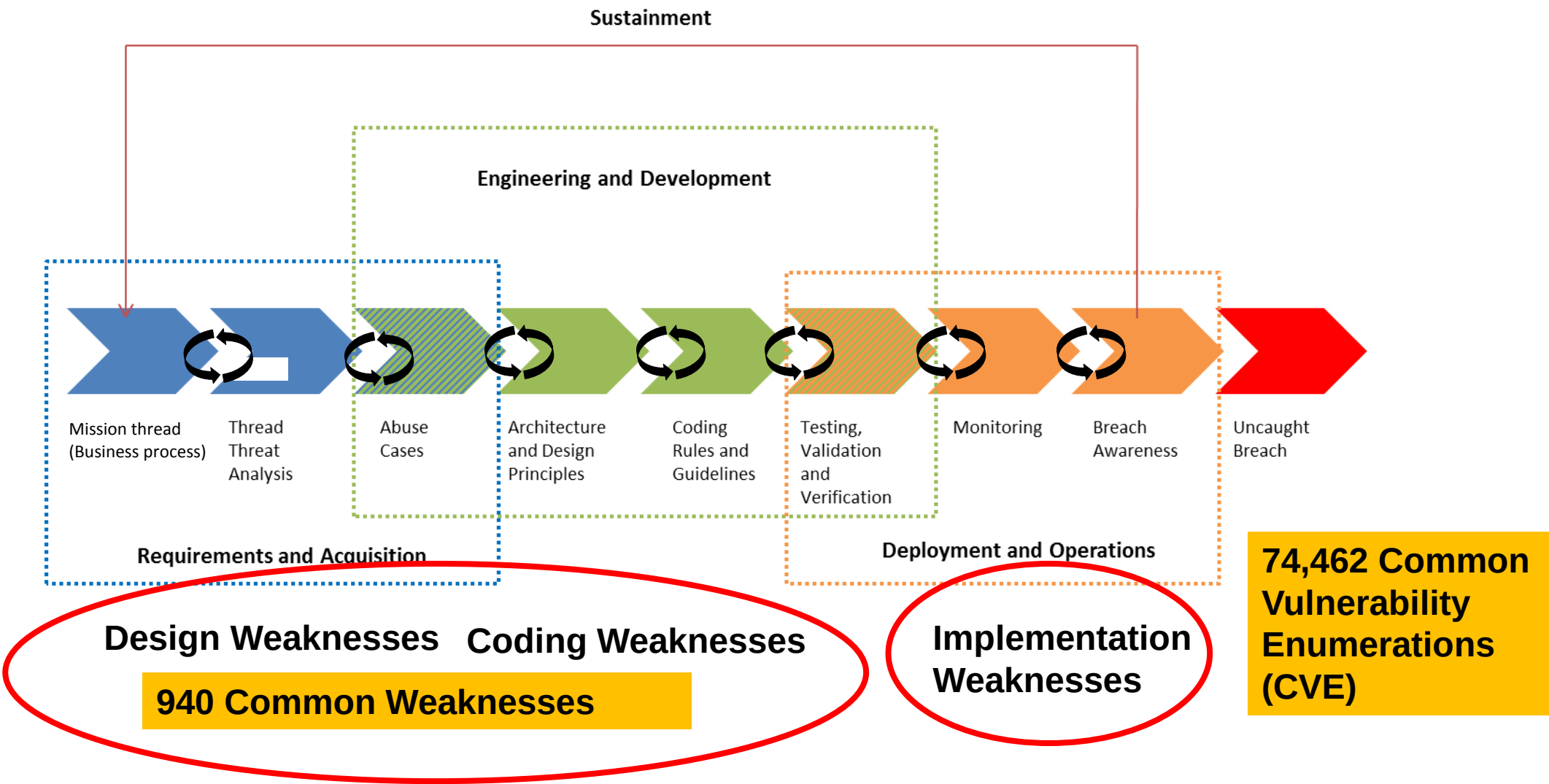
SwA for Development Cybersecurity Lifecycle



Cybersecurity Is a Lifecycle Challenge



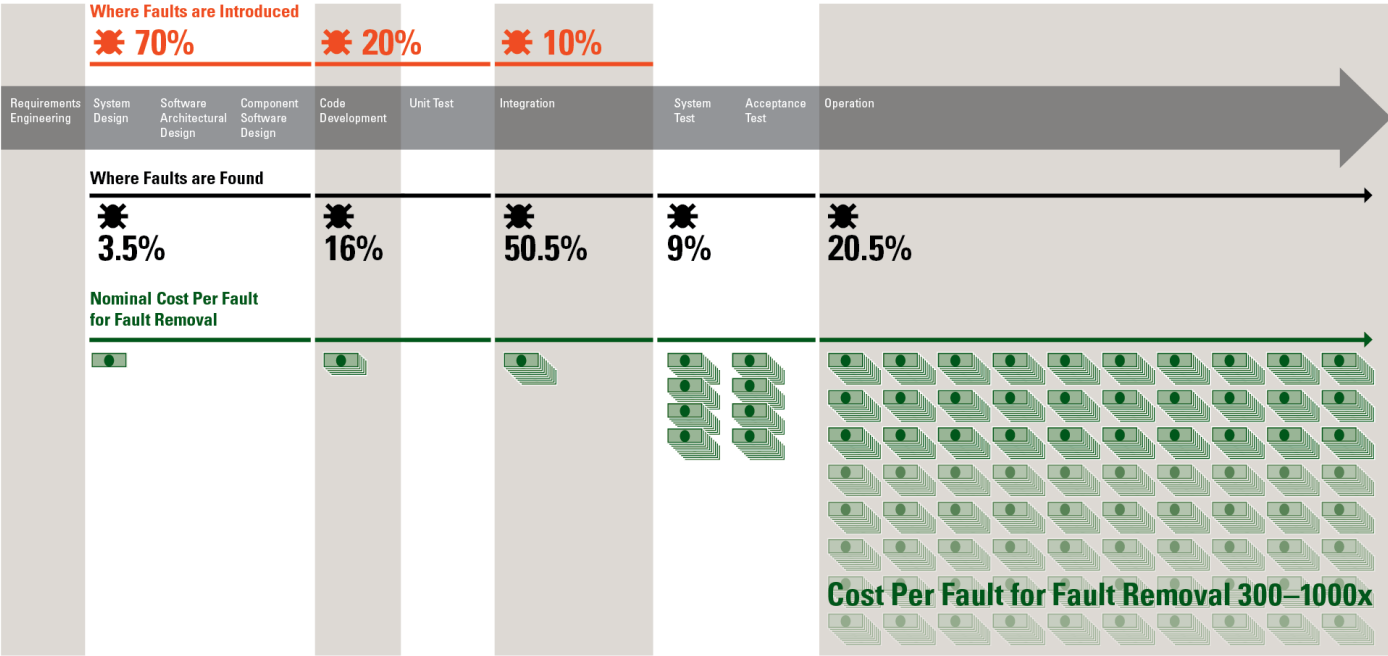
Cybersecurity Is a Lifecycle Challenge



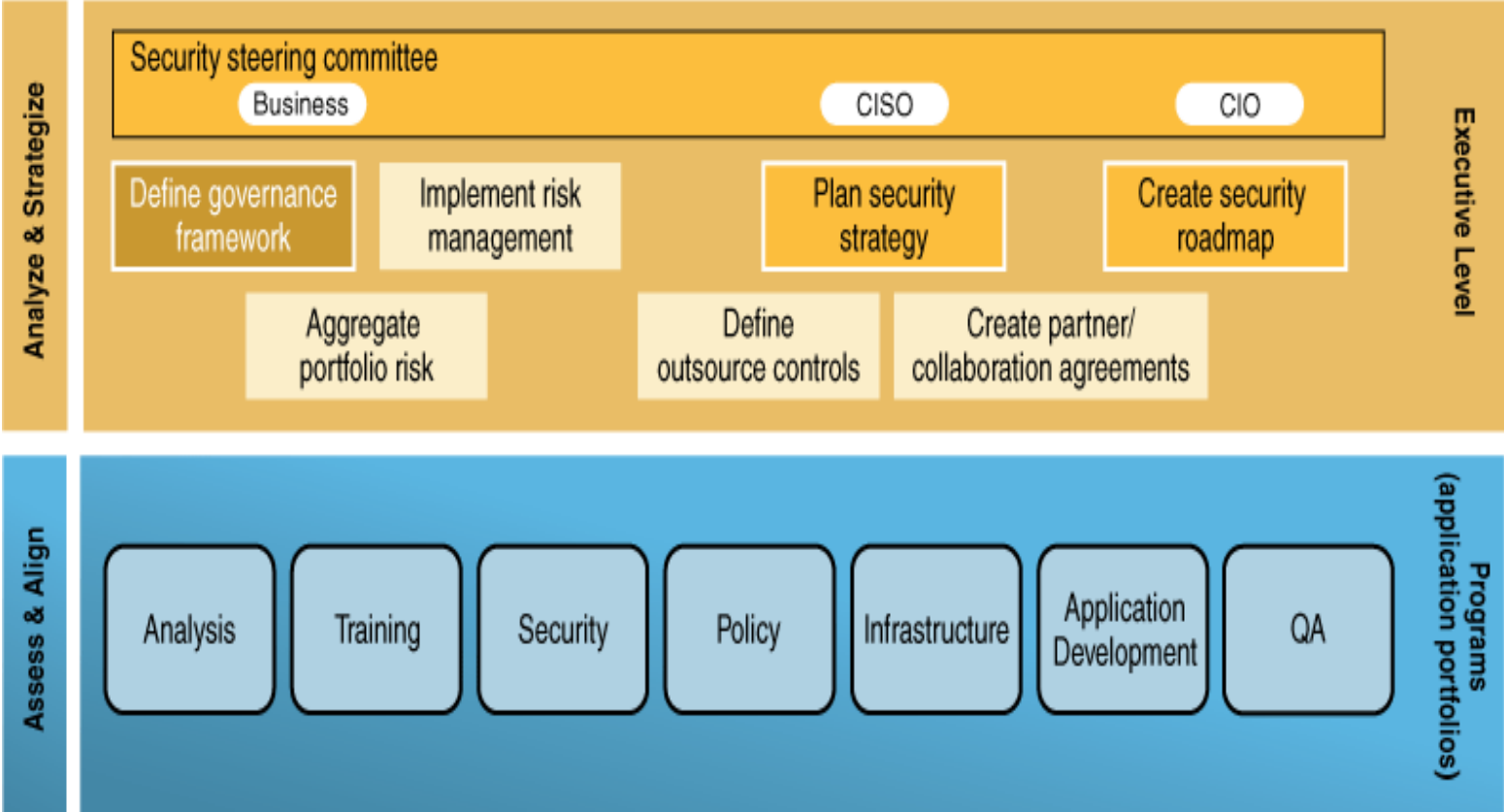
Faults account for 30–50% of total software project costs.

- Most faults are introduced before coding (~70%).
- Most faults are discovered at system integration or later (~80%).

Software Development Lifecycle

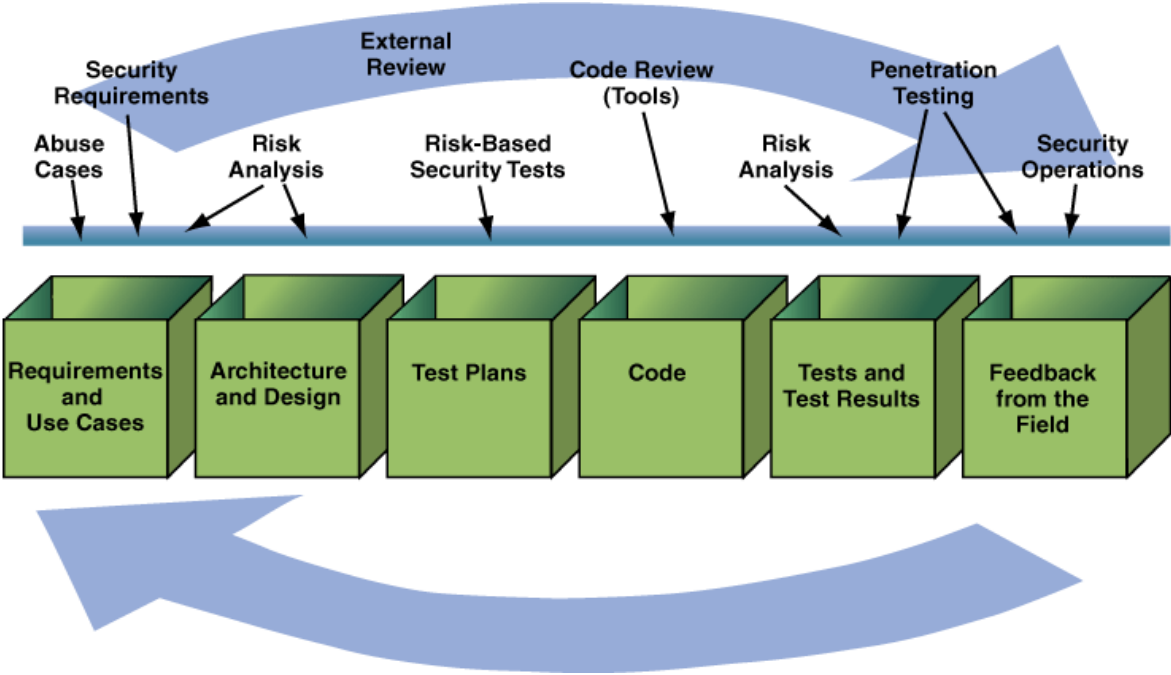


Enterprise Software Security Framework



Steven, John. "Adopting an Enterprise Software Security Framework." *IEEE Security & Privacy* 4, 2 (March/April 2006): 84–87.
<https://buildsecurityin.us-cert.gov/daisy/bsi/resources/published/series/bsi-ieee/568.html>

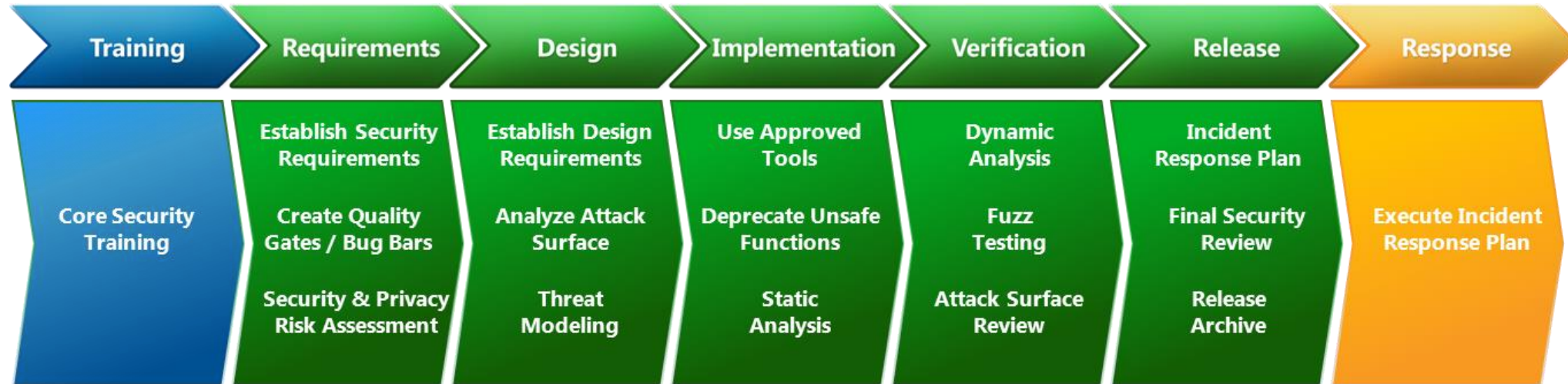
SDLC with Defined Security Touchpoints



SDLC: Software Development Life Cycle
McGraw, Gary. *Software Security: Building Security In*. Boston, MA: Addison-Wesley Professional, 2006.

Microsoft Security Development Lifecycle (MS SDL)

The Security Development Lifecycle (SDL) is a software development security assurance process consisting of security practices grouped by seven phases.

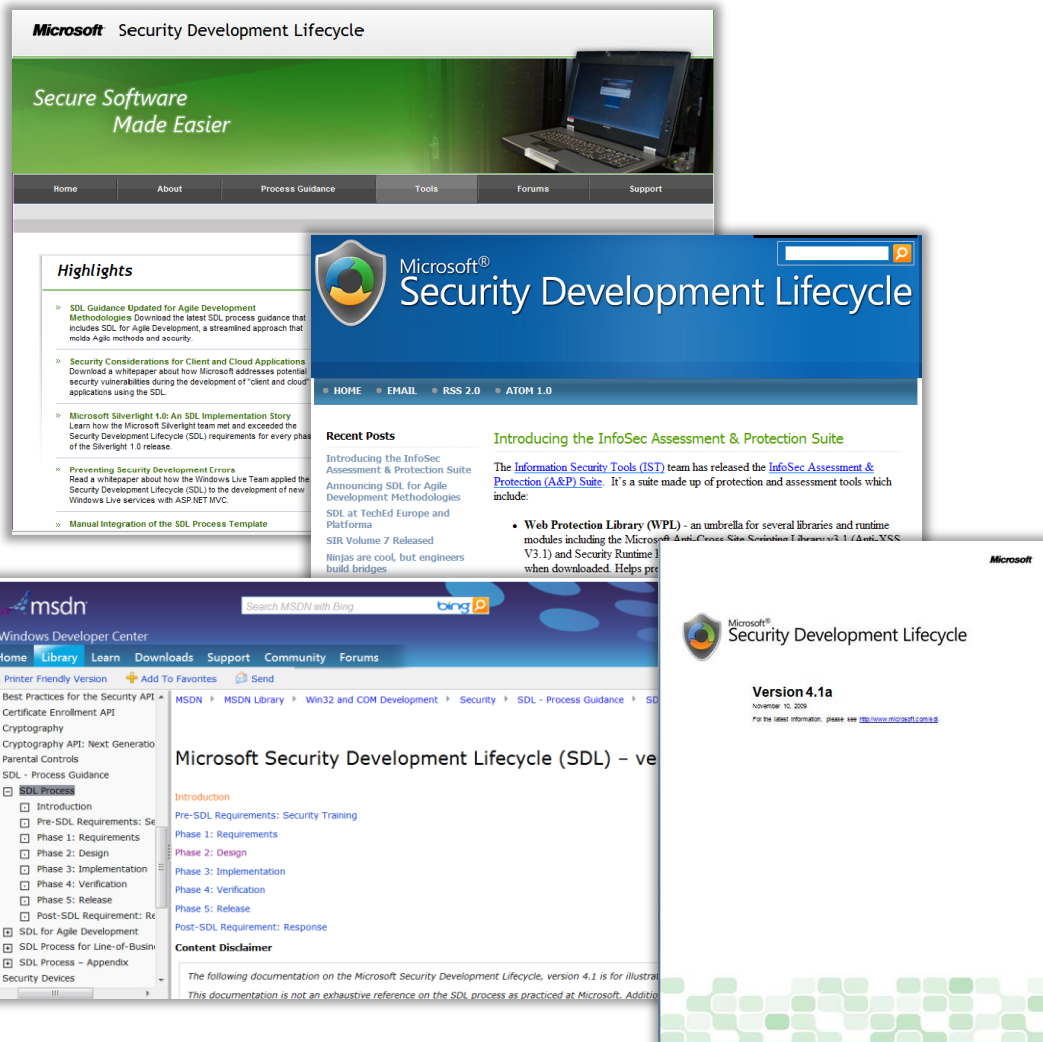


Reference: <http://www.microsoft.com/security/sdl/learn/measurable.aspx>

CERT Secure Practices Mapped to MS SDL

http://www.cert.org/archive/pdf/MS_CERT_SDL.pdf

MS SDL Resources



SDL Portal

<http://www.microsoft.com/sdl>

SDL Blog

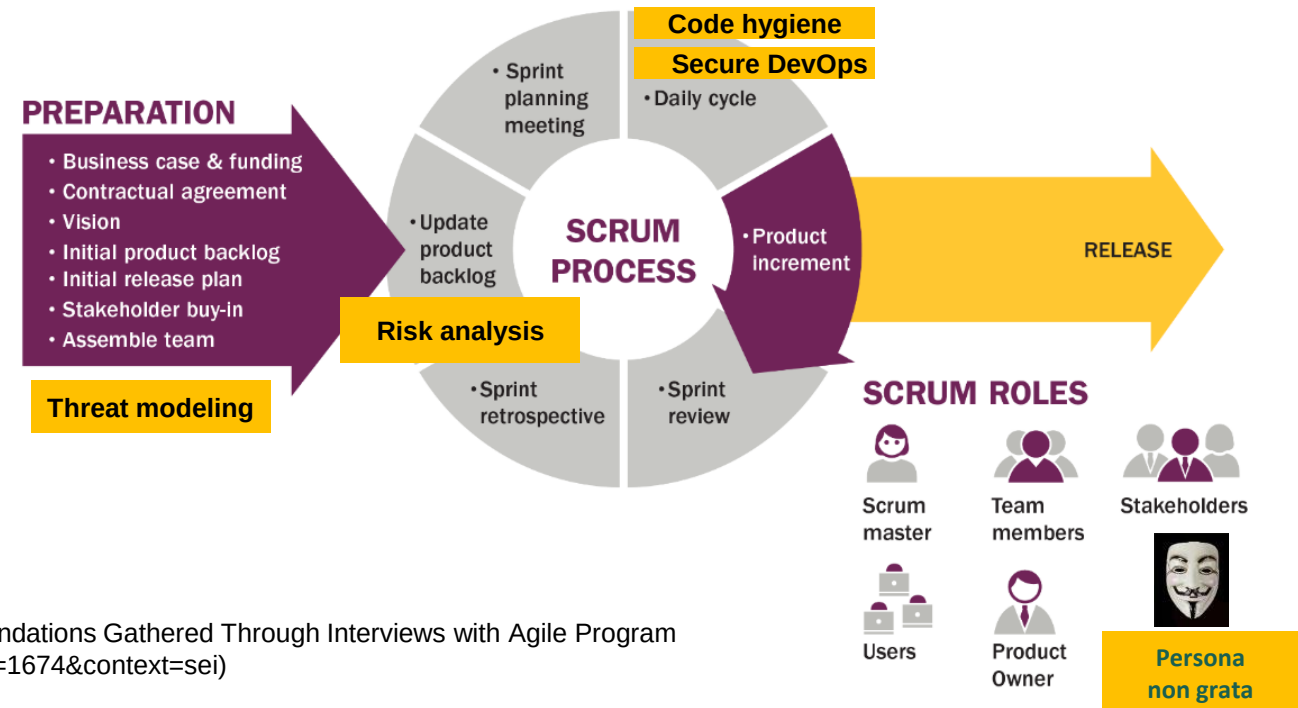
<https://blogs.microsoft.com/microsoftsecure/category/cybersecurity/security-development/>

SDL Process on MSDN (Web)

<http://msdn.microsoft.com/en-us/library/cc307748.aspx>

Integrating Security into Agile (Scrum) Development

1. Code hygiene – introduce secure coding
2. Secure DevOps – include security tools
3. Threat modeling – represent a new role
4. Risk analysis – prioritize in backlog



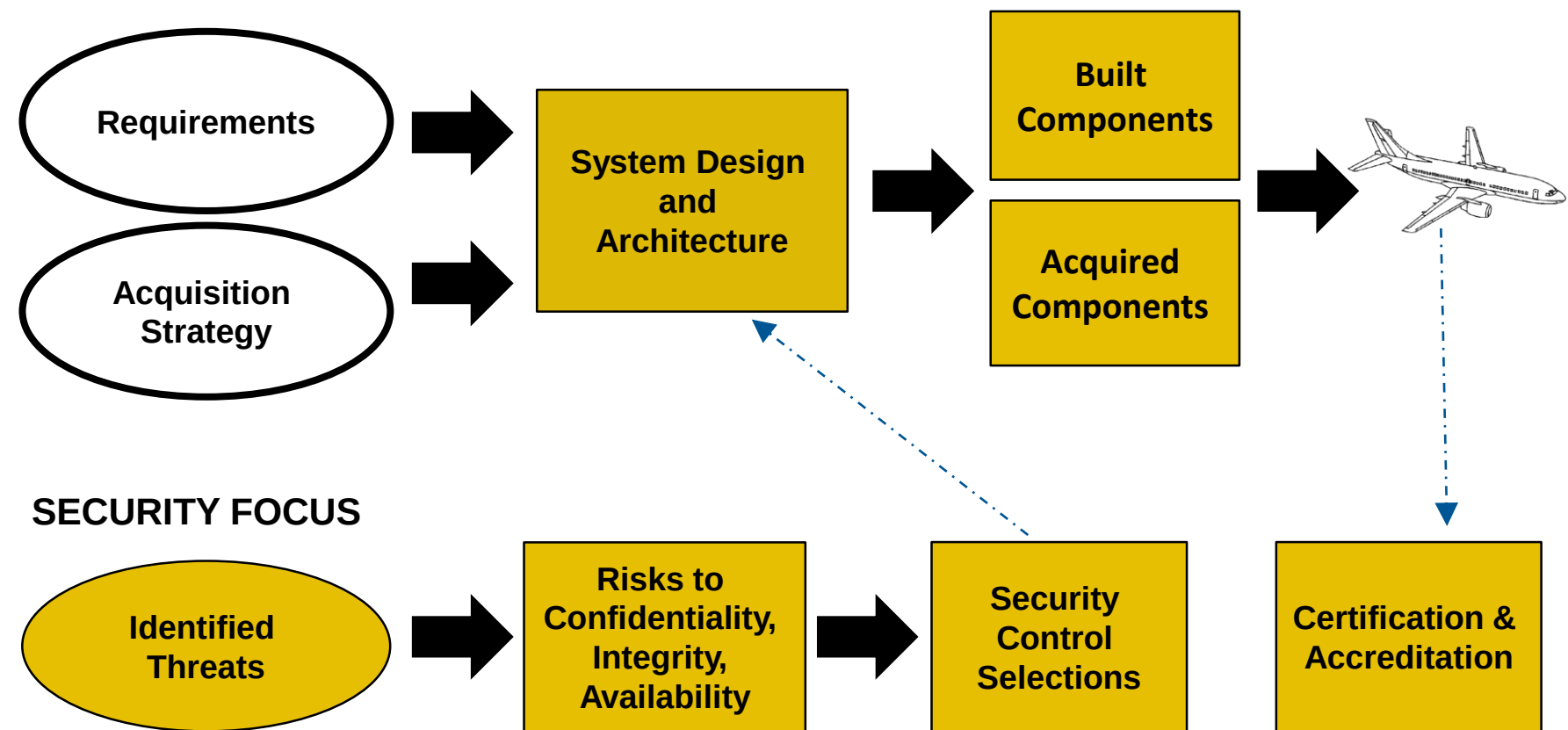
(See also: Bellomo and Woody, DoD Information Assurance and Agile: Challenges and Recommendations Gathered Through Interviews with Agile Program Managers and DoD Accreditation Reviewers (<http://repository.cmu.edu/cgi/viewcontent.cgi?article=1674&context=sei>))

SwA for Development

Cybersecurity Extends to the Organization



Building Security In: Is the System Secure?



How Do We Establish that the Plane Will Fly?



Planes are engineered to meet requirements that are defined for expected use.

Dreamliner Requirements

Carry passengers long distances in comfort

- 330 passengers; two-class configuration
- range: 11,910 km / 6,430 nmi
- wingspan: 197' 4"
- height: 55' 10"
- length: 224'



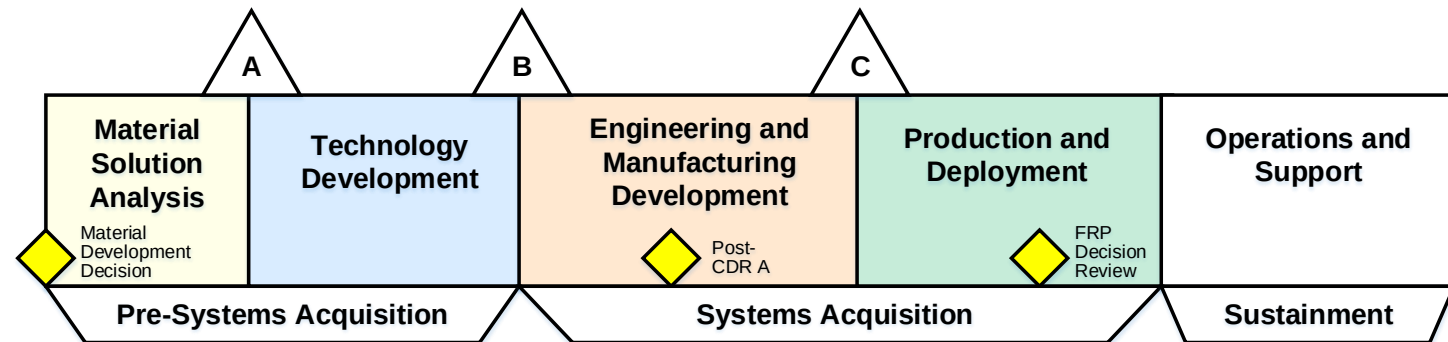
F35 Requirements

Single-seat, single-engine, all-weather stealth multirole

- range: approx. 1,620 nmi on internal fuel
- wingspan: 32.78'
- height: 13.33'
- length: 50.5'
- wing area: 450'
- empty weight: ~22,000 lbs.
- maximum speed: Mach 1.5 at altitude

DoD Acquisition Lifecycle

- Follow engineering best practices to define, monitor, and verify that requirements are met.
- Conduct milestone reviews to measure and confirm expected progress with the expected results.



Establish engineering requirements for software security.

Follow best engineering lifecycle practices for software security.

Collect evidence that engineering practices are being addressed (measurement).

Conduct milestone reviews to confirm expected progress with the expected results.



What

- Defines cybersecurity practices for acquiring and engineering software-reliant systems

Why

- Improve cybersecurity practices in acquisition programs

Benefits

- Provides the basis for assessing gaps in a program's cybersecurity practices and charting a course for improvement
- Establishes confidence in a program's ability to acquire software-reliant systems across the lifecycle and supply chain
- Reduces cybersecurity risk of deployed software-reliant systems

CMU/SEI-2017-TN-001: *SAF Introduction and Overview*
<http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=496134>



Process Management

- Process Definition
- Infrastructure Standards
- Resources
- Training

Engineering

- Product Risk Management
- Requirements
- Architecture
- Implementation
- Testing, Validation, and Verification
- Support Documentation and Tools
- Deployment

Project Management

- Project Plans
- Project Infrastructure
- Project Monitoring
- Project Risk Management
- Supplier Management

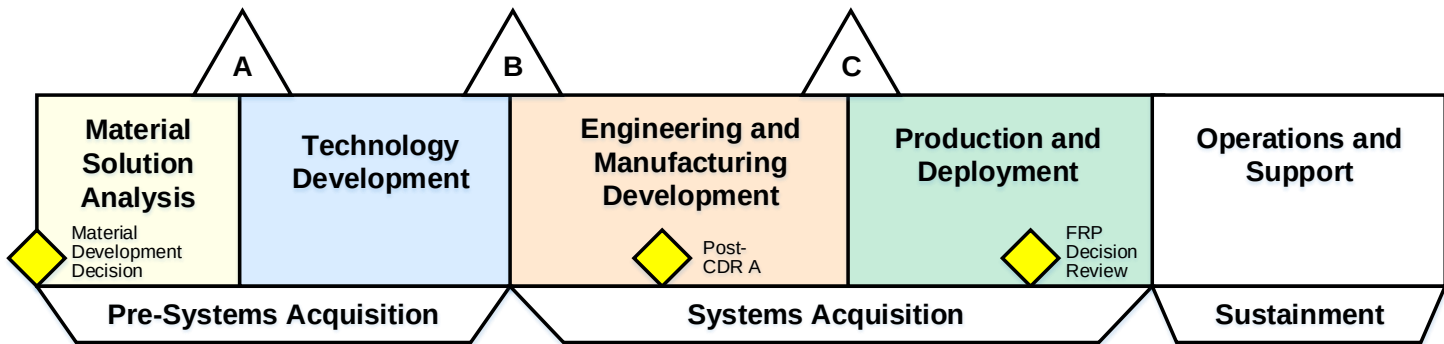
Support

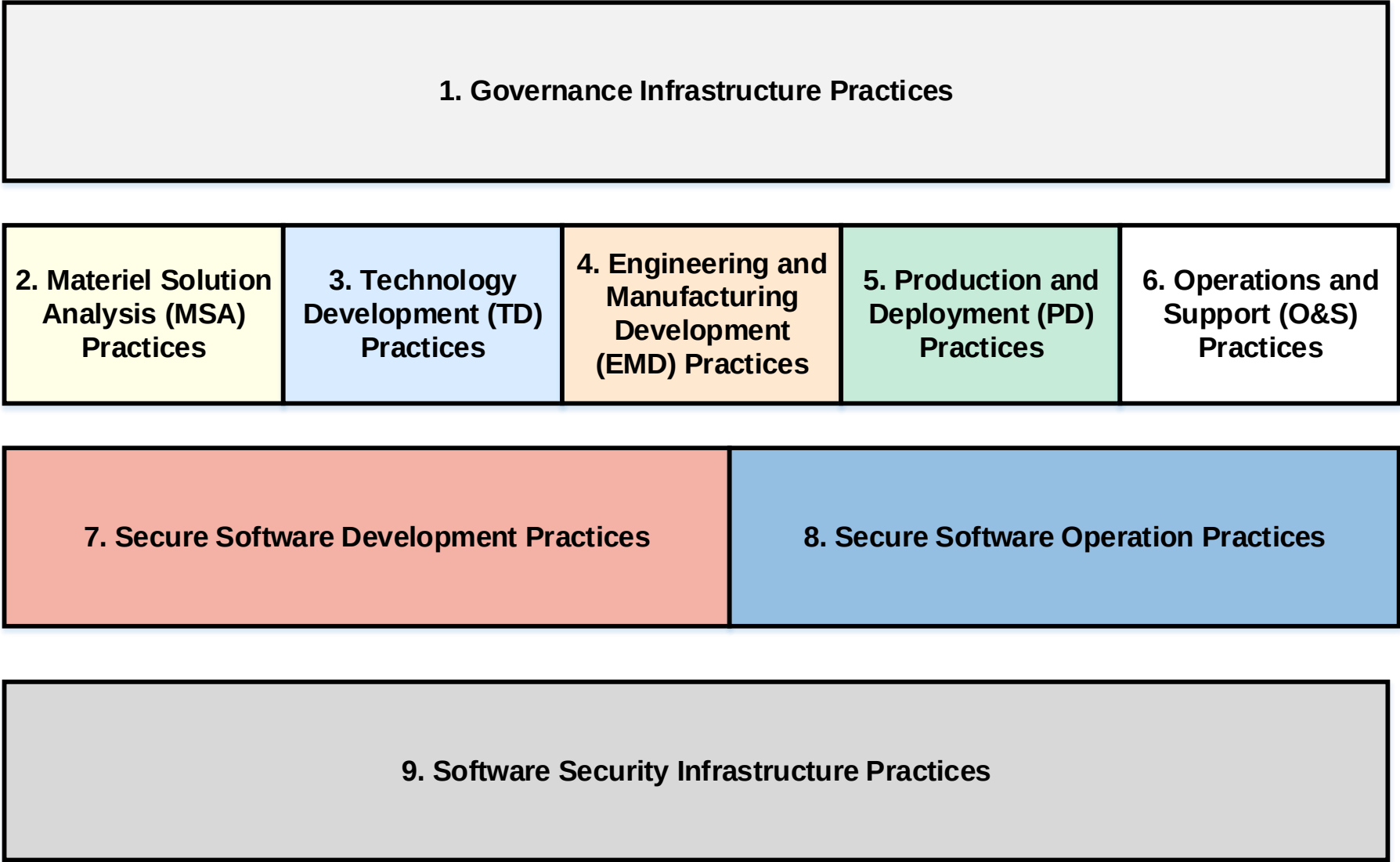
- Measurement and Analysis
- Change Management
- Product Operation and Sustainment

- **Requirements.** Does the program/project define and manage software security requirements?
- **Architecture.** Does the program/project appropriately address security in its software architecture and design?
- **Implementation.** Does the program/project minimize the number of vulnerabilities inserted into the code?
- **Testing, Validation, and Verification.** Does the program/project test, validate, and verify security in its software components?
- **Support Tools and Documentation.** Does the program/project develop tools and documentation to support secure configuration and operation of software components?
- **Deployment.** Does the program/project consider security during the deployment of software components?

DoD Acquisition Lifecycle

- Best practices for software assurance are mapped to the lifecycle.
- The SAF can be mapped to any acquisition/development lifecycle model.

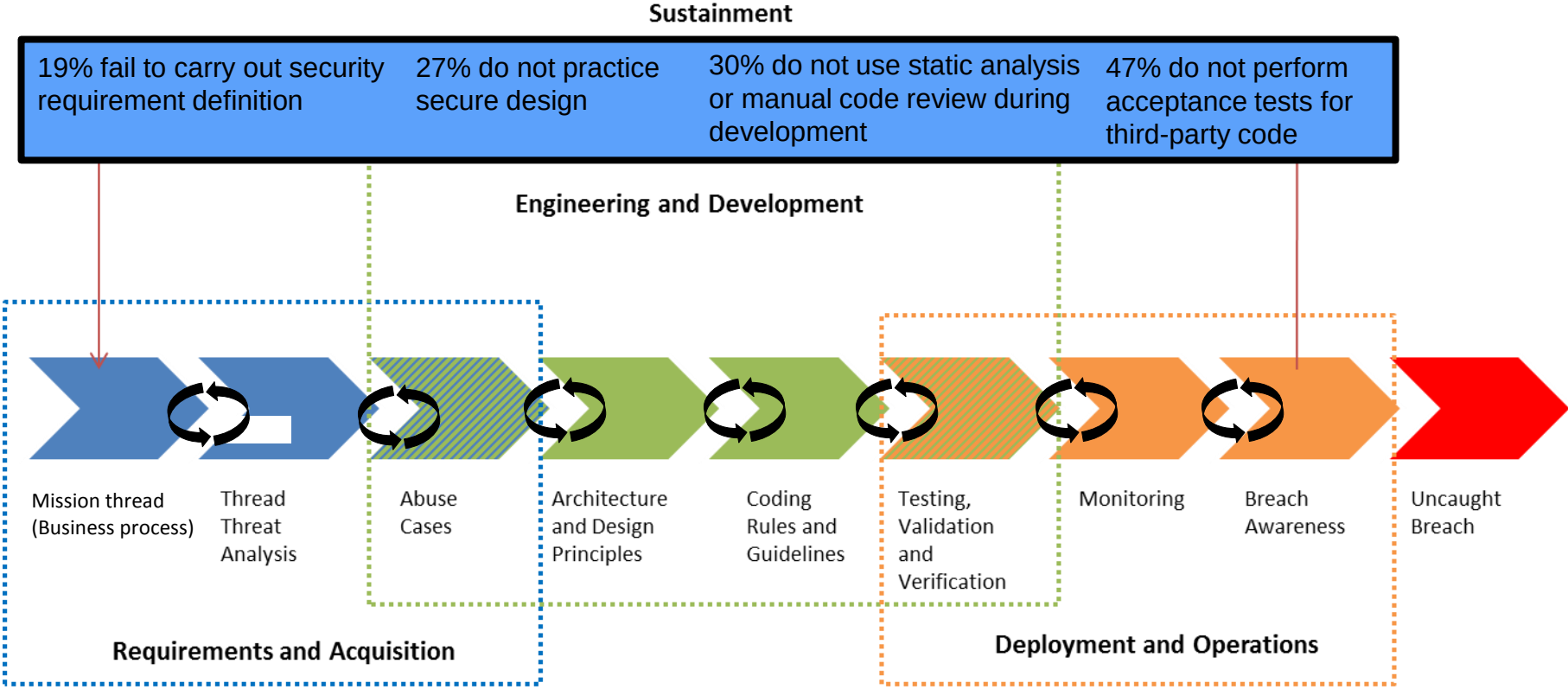






Does your organization have a lifecycle view of cybersecurity?
Is it integrated across the organization?

Opportunities for Improvement



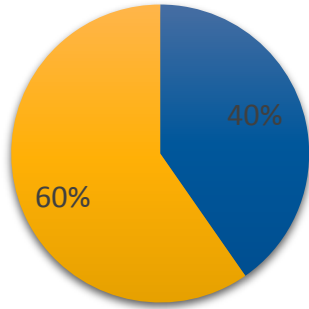
Less than 19% coordinate their security practices in various stages of the development lifecycle.

Source: Forrester Consulting, "State of Application Security," January 2011

SwA for Development Requirements Engineering Processes

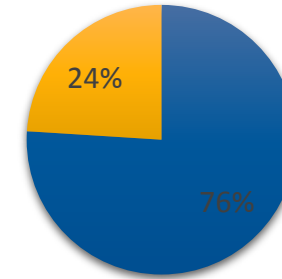


940 Total CWEs*



■ Design Weakness ■ Other Weakness

Top 25 CWEs (Most Dangerous)



■ Design Weakness ■ Other Weakness

Causes for design weaknesses:

- **Poor security requirements**
- Limited understanding of the impact of security risk on mission success

Source: <http://cwe.mitre.org/> as of Feb 9, 2014

A capability needed by a user to solve a problem or achieve an objective

A capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document

The set of all requirements that form the basis for subsequent development of the software or software component

(Source: IEEE Software Requirements Engineering, Dorfman & Thayer)

Typical problems with security requirements

- Stated as specific security solutions (practices) and not real requirements
- Too narrowly focused on security in a particular application (e.g., use SSL for Web communication)
- Compliance mandates are substituted for security requirements
- Ignored in requirements elicitation because no stakeholders are knowledgeable enough about security impacts to state their security requirements
- Focused on selection of controls after designs are complete

Comprehensive, Lightweight, Application Security Process (CLASP)

Core Security Requirements Artifacts

Security Quality Requirements Engineering (SQUARE)

Security Requirements Engineering Process (SREP) – based on common criteria

Security Patterns

What is the value of a focus on security requirements?

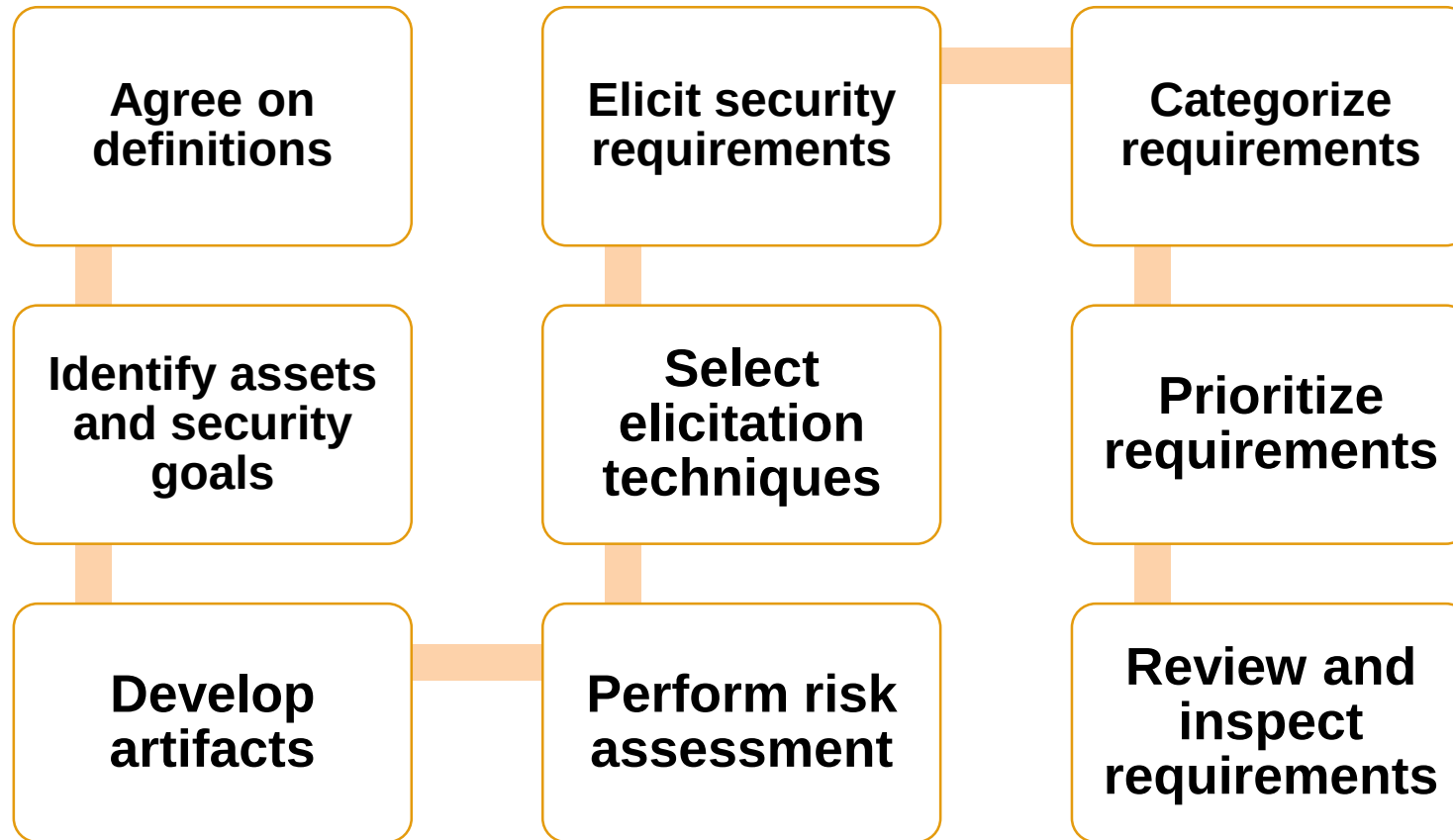
- **Requires different stakeholder and expert participation**
- **Security requirements information should also be protected (informs an attacker)**
- **Focus is on how the system could break and what the system should not do**

Methodology for eliciting, categorizing, and prioritizing security requirements for information technology systems and applications

In SQUARE, security requirements are:

- defined as system and software quality attributes
- structured to integrate with the system's functional requirements
- consistent with system and software requirements engineering and practices

The SQUARE Process



Materials and tool at <http://www.cert.org/sse/square/>

Several versions:

- SQUARE- Lite
- SQUARE for Privacy
- SQUARE for Acquisition

Can be integrated with an existing requirements engineering processes to address gaps

Materials and tool at <http://www.cert.org/sse/square/>

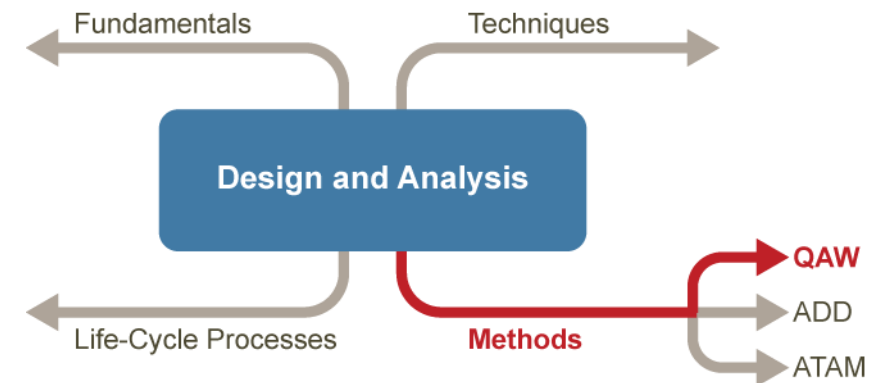
SwA for Development Assured Design



Quality Attribute Workshop (QAW) - facilitated method that engages system stakeholders early in the lifecycle to discover the driving quality attribute requirements of a software-reliant system

Architecture Tradeoff Analysis Method (ATAM) - method for evaluating software architectures relative to quality attribute goals to expose architectural risks that potentially inhibit the achievement of an organization's business goals; provides insight into how those quality goals interact with each other—how they trade off against each other.

**Security,
Trustworthiness,
Reliability can be
selected as focus
qualities**

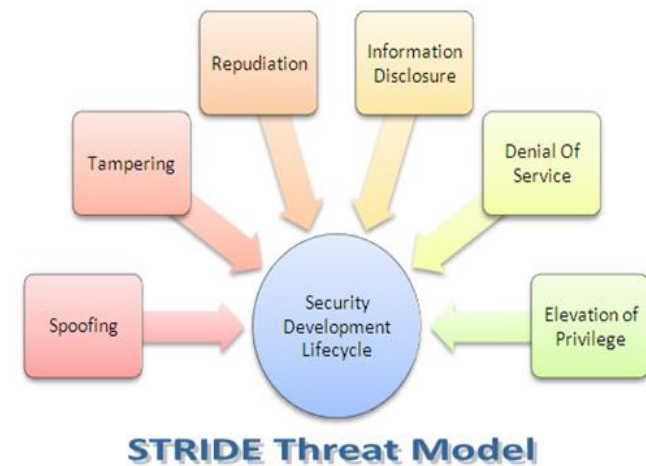


Using MS STRIDE to determine threats

The STRIDE approach to modeling threats was developed by researchers at Microsoft.

The name STRIDE is an acronym based on the initials of the threat categories: Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege.

This is a way to walk through the categories of threats and determine the potential consequences of each.



Model the Attack Surface: Diagram Elements

External Entity

- People
- Other systems
- Microsoft.com

Process

- DLLs
- EXEs
- COM object
- Components
- Services
- Web Services
- Assemblies

Data Flow

- Function call
- Network traffic
- Remote Procedure Call (RPC)

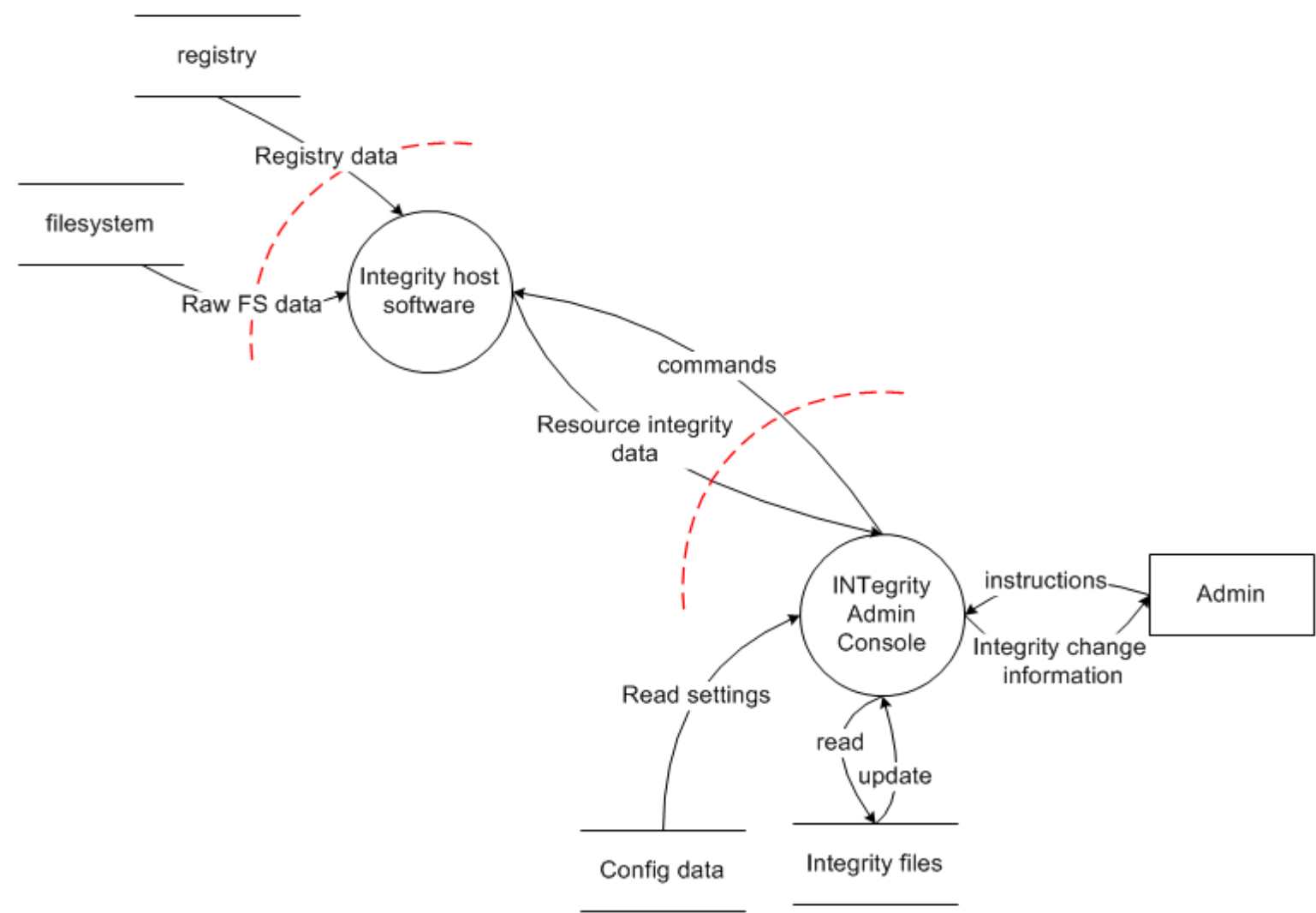
Data Store

- Database
- File
- Registry
- Shared Memory
- Queue / Stack

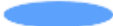
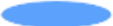
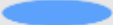
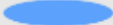
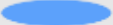
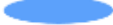
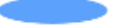
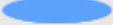
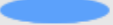
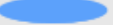
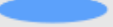
Trust Boundary

- Process boundary
- File system

Example – Level 1 Diagram



The STRIDE categories each have different architectural implications.

Element	Interactors	Data flows	Data stores	Processes-Software
Spoofing identify				
Tampering with data				
Repudiation				
Information disclosure				
Denial of service				
Elevation of privilege				

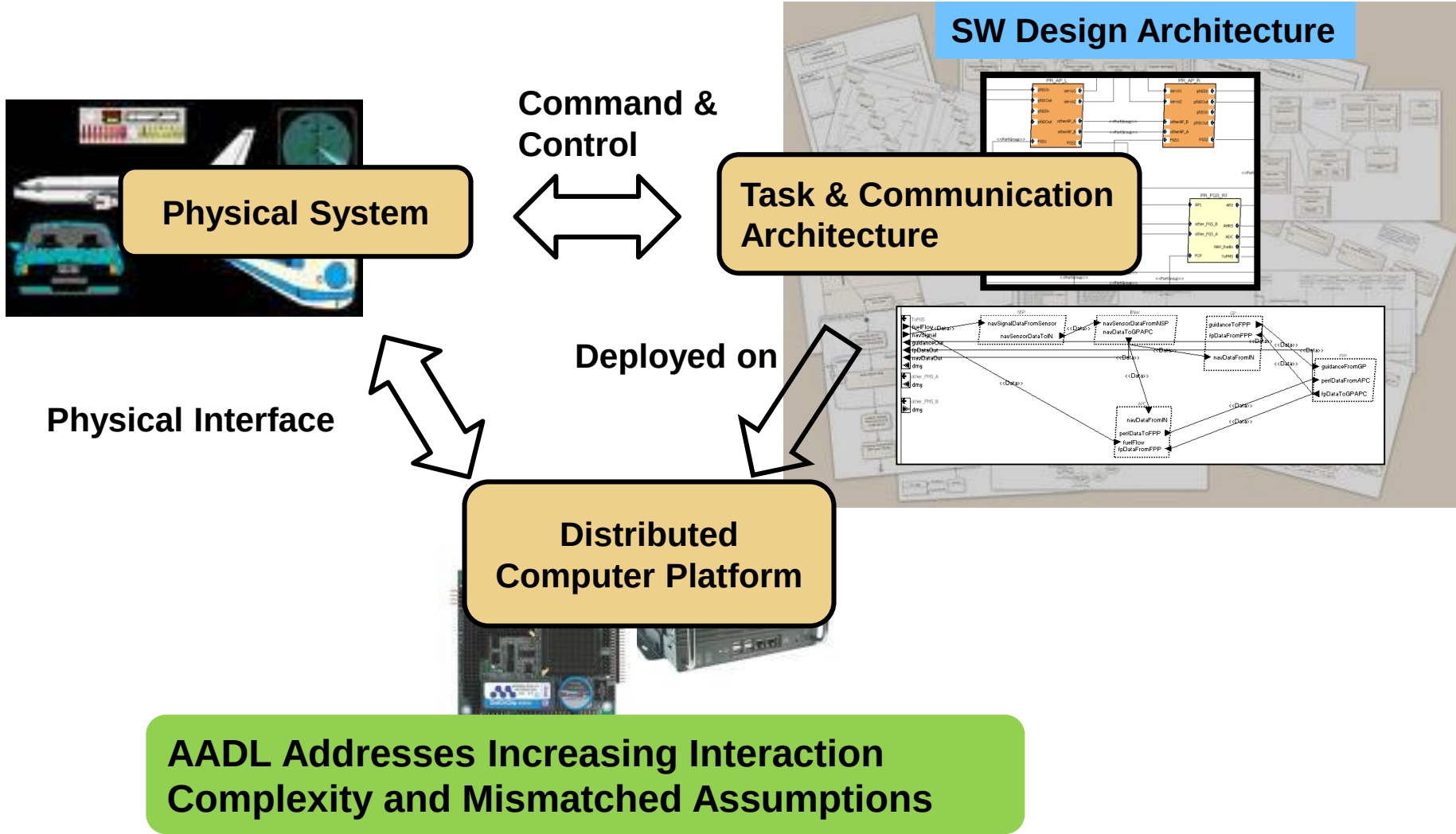
Standard Mitigations

Spoofing	Authentication	To authenticate principals: • Cookie authentication • Kerberos authentication • PKI systems such as SSL/TLS and certificates To authenticate code or data: • Digital signatures
Tampering	Integrity	• Windows Mandatory Integrity Controls • ACLs • Digital signatures
Repudiation	Non Repudiation	• Secure logging and auditing • Digital signatures
Information Disclosure	Confidentiality	• Encryption • ACLs
Denial of Service	Availability	• ACLs • Filtering • Quotas
Elevation of Privilege	Authorization	• ACLs • Group or role membership • Privilege ownership • Input validation

Attack Surface Tool download: <https://www.microsoft.com/en-us/download/details.aspx?id=24487>

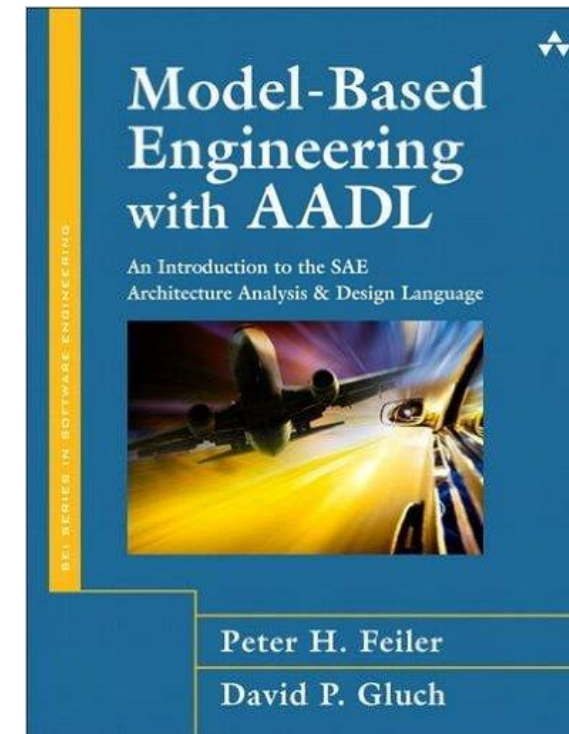
Related Discussion of Microsoft SDL:
<https://www.microsoft.com/en-us/sdl/default.aspx>

Additional Microsoft Tool Downloads:
<http://www.microsoft.com/security/sdl/adopt/tools.aspx>



AADL (Architecture Analysis and Design Language):

- specifies a static representation of a system architecture
- can model logical flows, binding of software to hardware
- is strongly typed, allowing a consistency checks via a modeling tool set.



We evaluated security concepts and associated semantics and behavior to explore what extension mechanisms would suffice to allow specification and analysis of security concerns.

To drive this analysis we needed concrete threats to analyze in the context of a system which required a specific example

(e.g., automotive electronics)

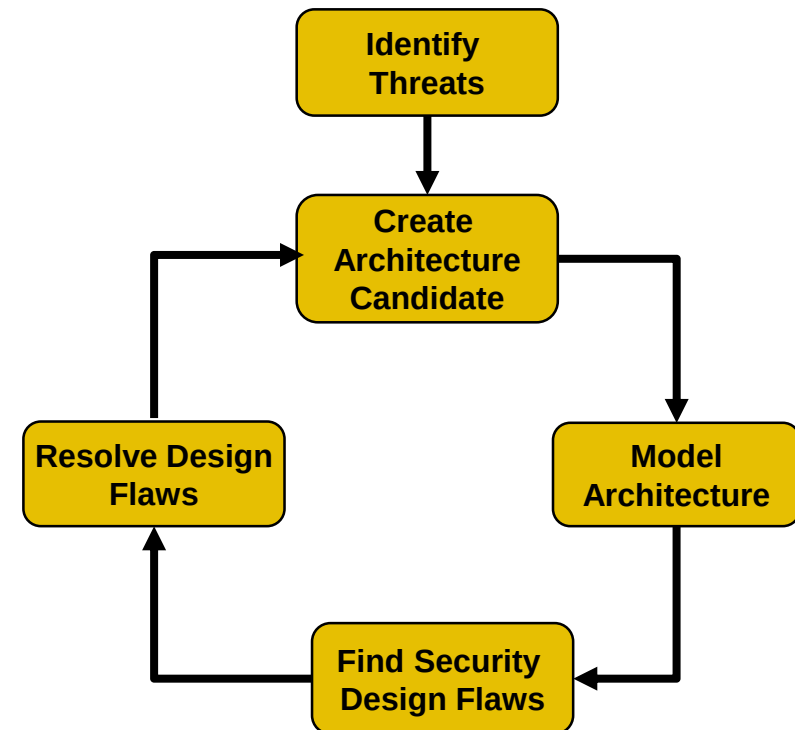


Ellison, Robert et al. *Extending AADL for Security Design Assurance of Cyber-Physical Systems*. CMU/SEI-2015-TR-014. Software Engineering Institute, Carnegie Mellon University. 2015. <http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=449510>

Building security in means that security requirements are abstracted into the design characteristics and constraints.

Crucial security decisions to address threats are made in the architecture.

Formal modeling provides a means to continuously verify an architectural design as new threats are considered.



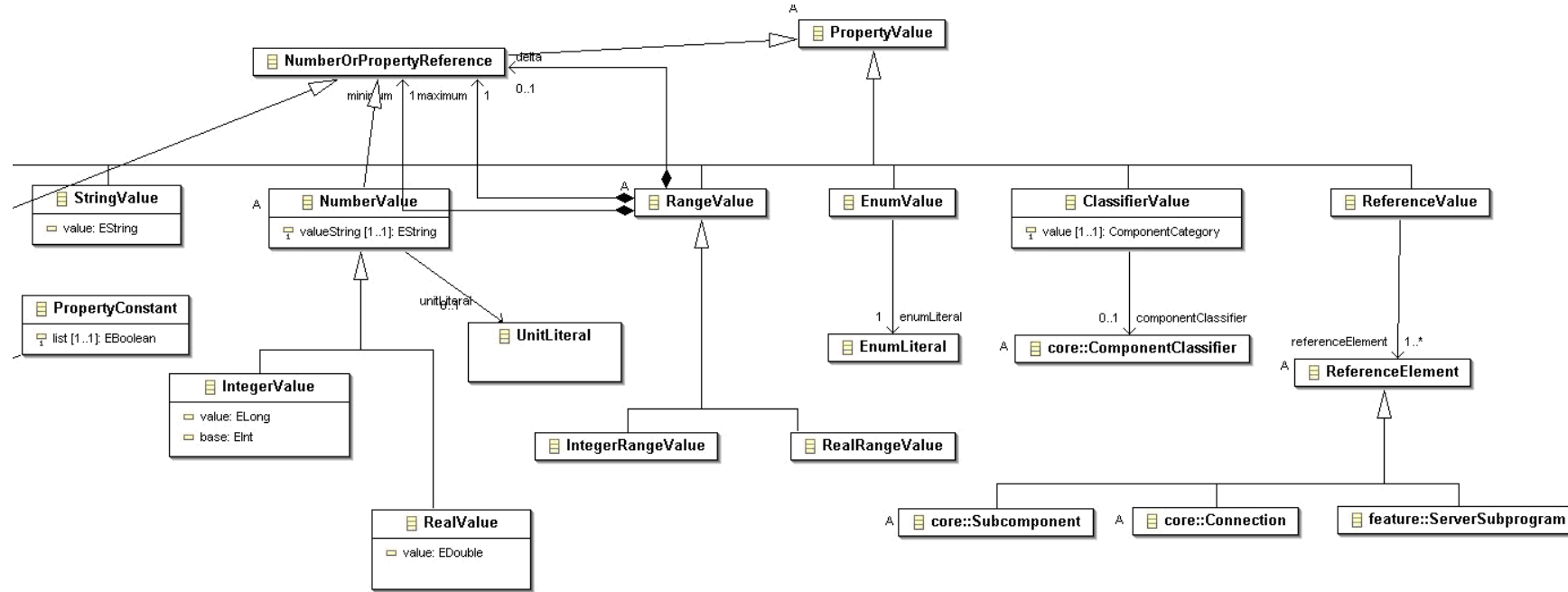
Determine Threat Requirements Before Modeling

We instantiated STRIDE in the context of the automotive electronics example. For each threat category we described:

1. how the threat might be realized: the conditions under which it might occur (e.g., Infotainment system could be used as an attack entry point to control car speed)
2. the implications of this threat on the infotainment system and, in particular, how the system is represented in an AADL model
3. the risks that arose from a consideration of the threat in the infotainment context
4. questions about the architecture of the infotainment system, and its implementation, that will inform the analysis
5. the security property that is violated if this threat is not mitigated

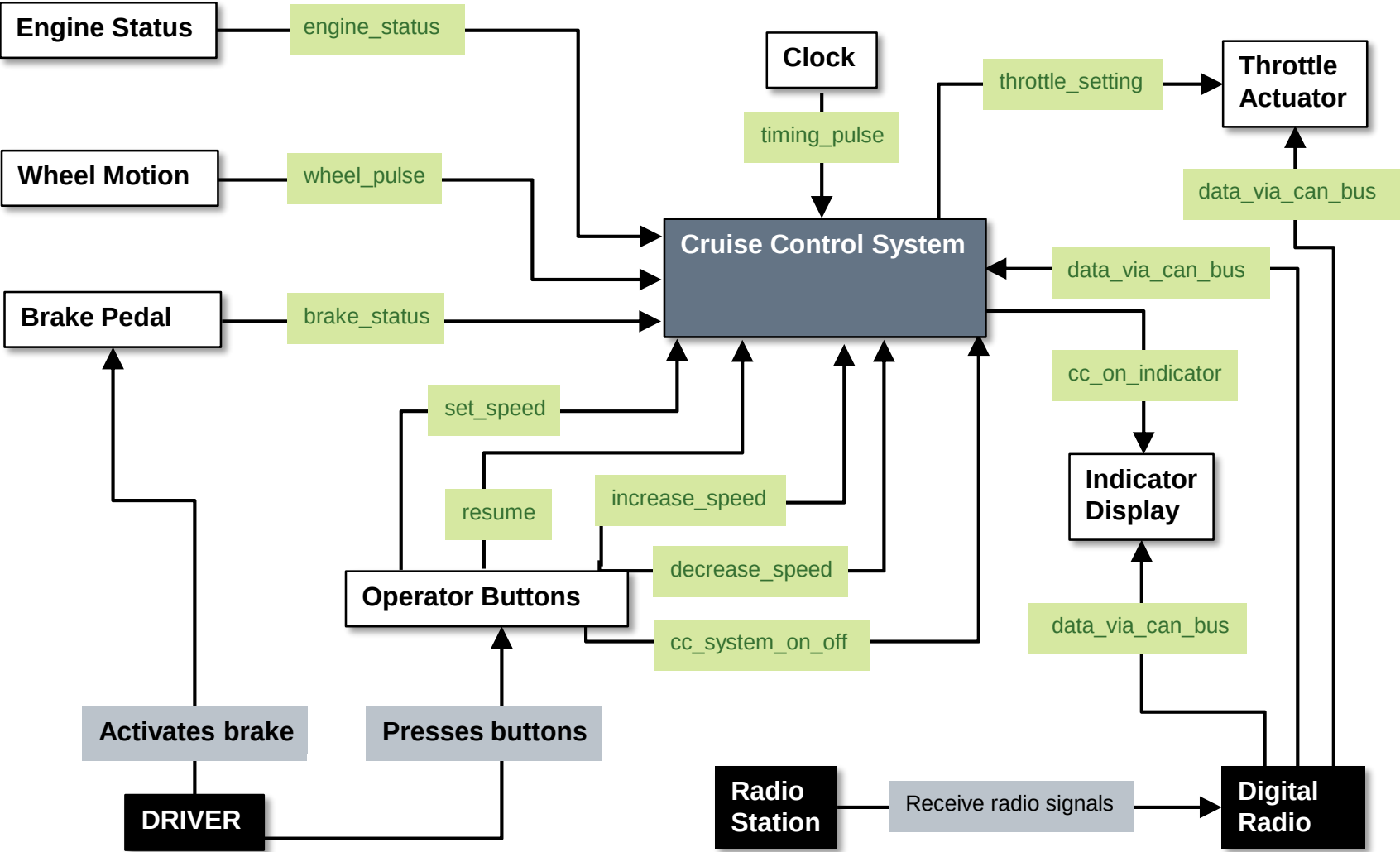
AADL for Modeling Security Properties

AADL has *properties* to capture component characteristics.



Example: a component that performs encryption can have one property for latency (e.g., 3 ms.), and another for type of encryption (e.g., RSA.)

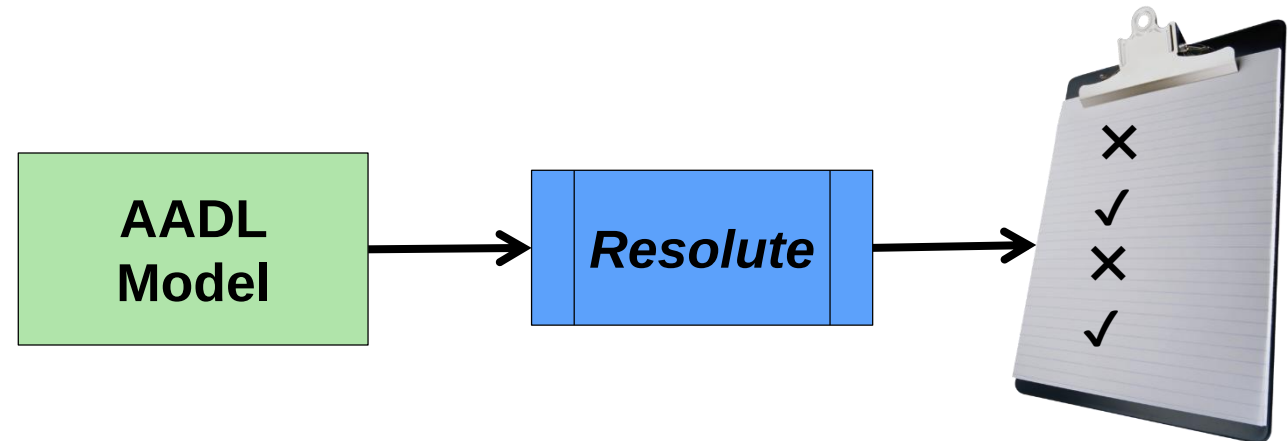
Modeling the Automotive Electronics System



To reason about the satisfaction of security properties in a system architecture we need to:

1. Specify the properties, and the architectural elements they are associated with in AADL; and then
2. Analyze claims over those properties.

To analyze claims (and verify changes to the model) we use the *Resolute* model-checker.



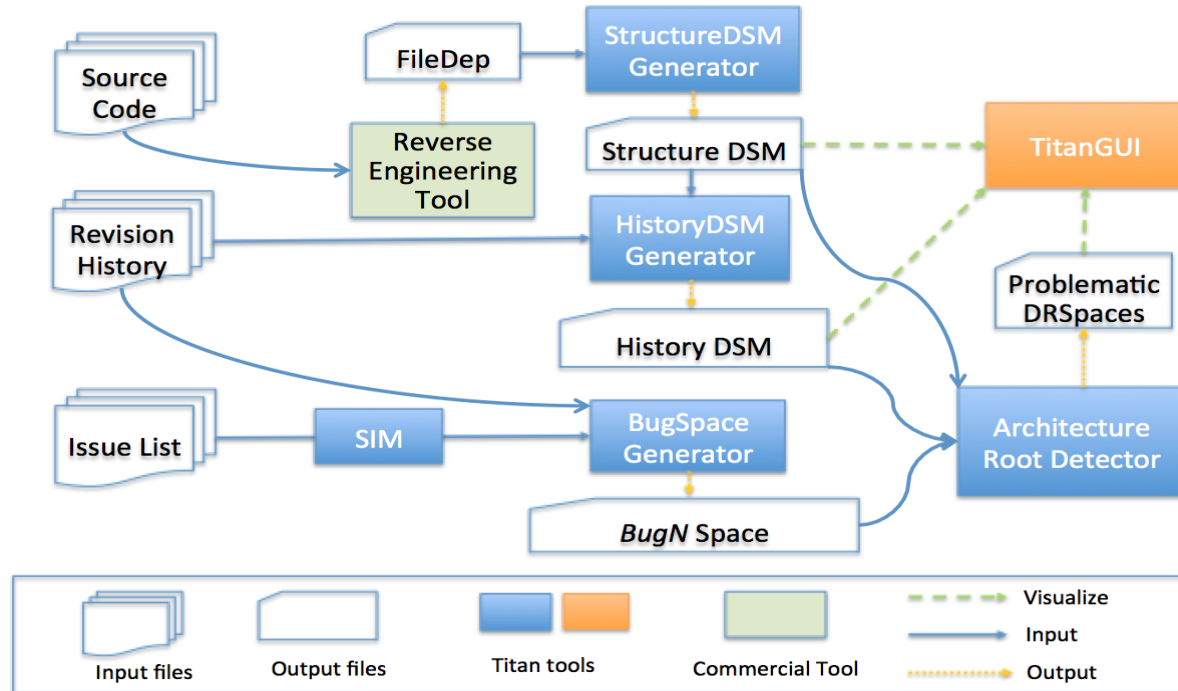
Architectural security modeling has several benefits:

- guides an architect to reason and design in terms of system-wide security properties
- provides a framework for checking that such properties are maintained during system maintenance and evolution

Limitations: does not fully support security *validation*. Is the system sufficiently secure for the planned usage?

- A weakness *external to the model* enables an attacker to operate outside the model. Example: A buffer overflow vulnerability or successful phishing can enable an attacker to obtain privileges without authentication.
- Specifications may be correct but not provide the desired level of security assurance. Example: An architect claims that a component sufficiently verifies input data to eliminate the risk of a SQL-injection. But CWE 89 recommends alternative mitigations with a higher level of assurance than input verification, such as component output verification.

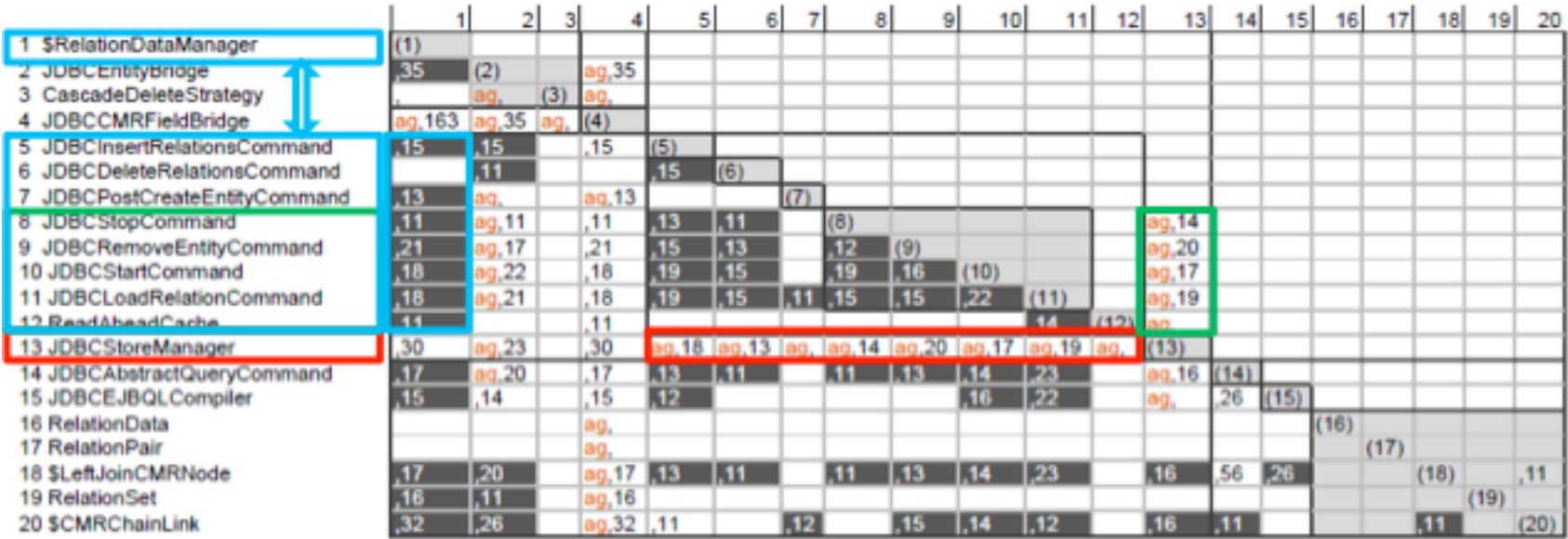
Architecture Design Rule Analysis for Security (ADRAS)



Research to identify and address security technical debt:
automated analysis of the architecture implemented in the code
to discover weaknesses using the Titan tool suite

- (1) Unstable Interface
- (2) Implicit Cross-Module Dependency
- (3) Unhealthy Inheritance Hierarchy
- (4) Cross-Module Cycle,
- (5) Cross-Package Cycle.

Analyzing the Architectures of Existing Systems with Design Rule Spaces (DRPaces)



Design rules are the critical architectural decisions that decouple the rest of the system into independent modules

Use relationships identified in the code to identify design patterns and faults

J. Ryoo, P. Anand, R. Kazman, "Architectural Analysis of Security Vulnerabilities", *IEEE Security and Privacy*, November/December 2015.

R. Mo, Y. Cai, R. Kazman, L. Xiao, "Hotspot Patterns: The Formal Definition and Automatic Detection of Architecture Smells", *Proceedings of The Working IEEE/IFIP Conference on Software Architecture (WICSA 2015)*, (Montreal, Canada), May 2015.

- If the leading class of a DRSpace is error-prone, the files in the DRSpace also tend to be error-prone.
- The majority of the error-prone files in a project are concentrated in just a few DRSpaces (<10%).
- By combining evolution and structure coupling, we can determine not just the locations of errors, but also the reasons for them.
- If a file is involved in greater numbers of architecture flaws, it is more error-prone/change-prone than average files.

https://insights.sei.cmu.edu/sei_blog/2016/02/a-tool-to-address-cybersecurity-vulnerabilities-through-design.html



Collected wisdom from thousands of contributors on community wiki since Spring 2006

SEI CERT C Coding Standard

- Free PDF download:

<http://cert.org/secure-coding/products-services/secure-coding-download.cfm>

- Basis for ISO TS 17961 C Secure Coding Rules



SEI CERT C++ Coding Standard

- Free PDF download (Released March 2017):

<http://cert.org/secure-coding/products-services/secure-coding-cpp-download-2016.cfm>

CERT Oracle Secure Coding Standard for Java

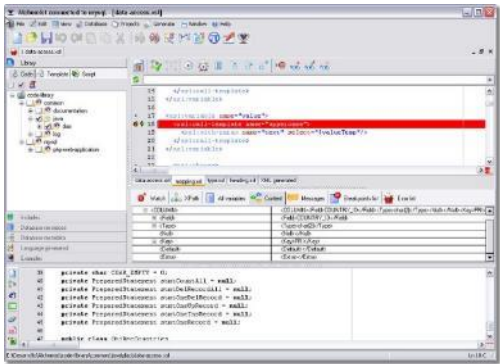
“Current” guidelines available on CERT Secure Coding wiki

- <https://www.securecoding.cert.org>

Adoption of Secure Coding Standards

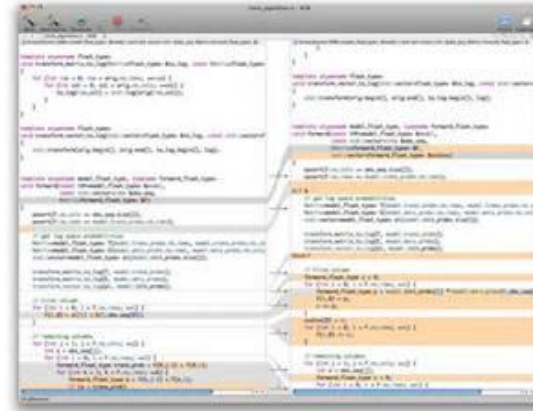


Training



Integrated development environments (IDE)

Tools Encourage Application of Secure Coding

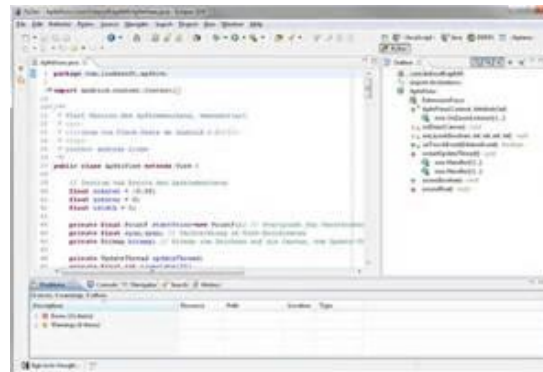


Moving rules into IDE improves application of secure coding

- Early feedback corrects errors on introduction
- Exceptions are understood in context
- Feedback improves developer skill

Target Clang static analyzer (C based languages)

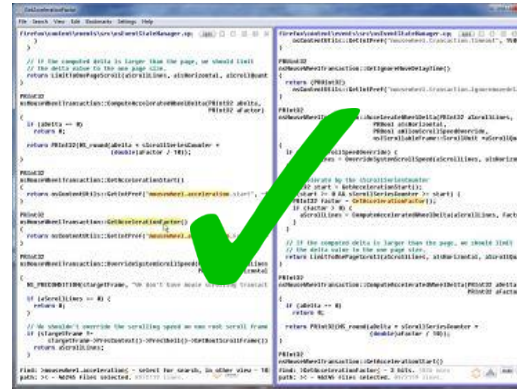
- Widely used open source front end for popular compilers
- Integrated into Apple's Xcode IDE



Target FindBugs (Java)

- Integrated into Eclipse and JDeveloper

Use Static Testing and Source Code Analysis



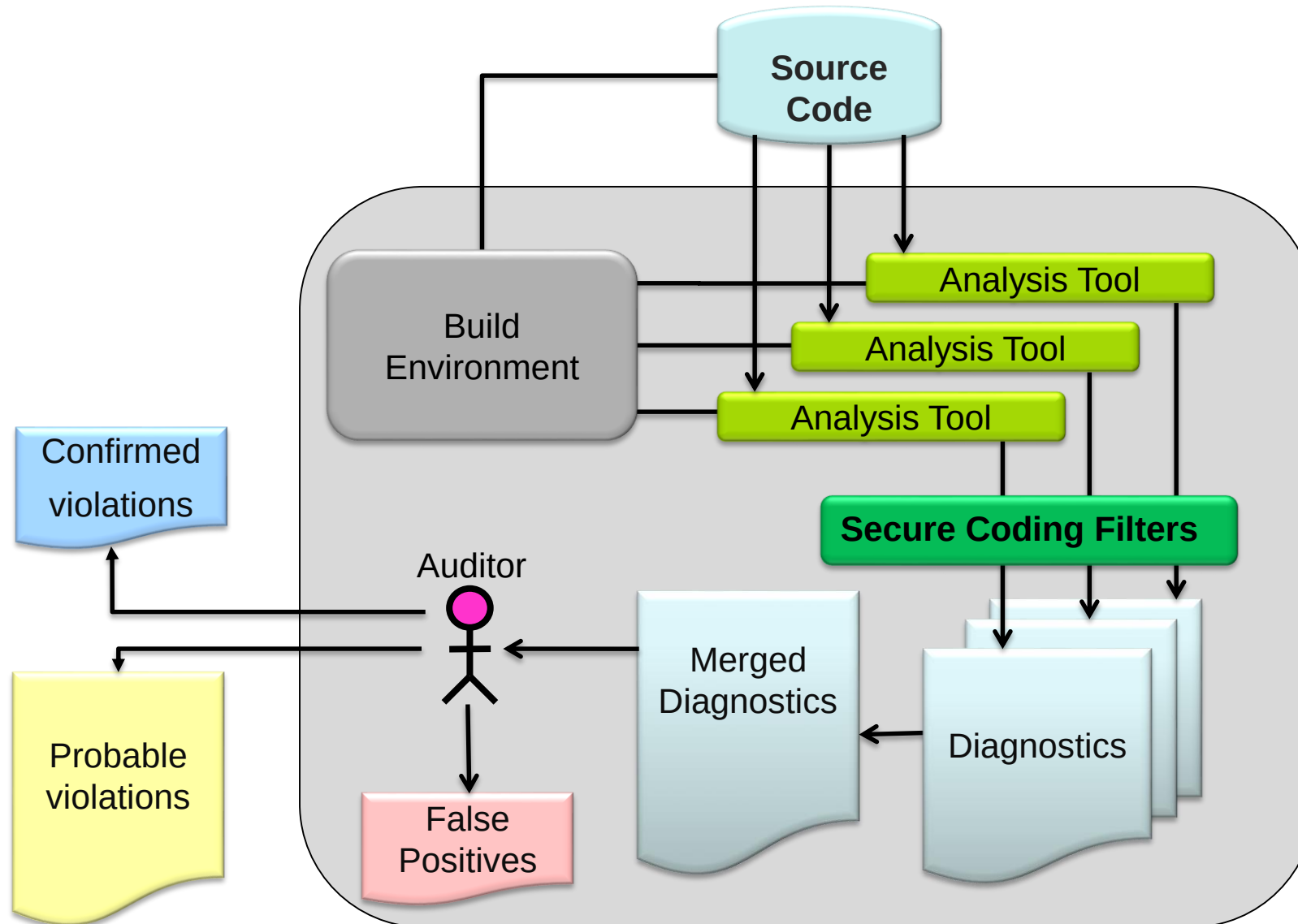
Secure Code Analysis Laboratory (SCALE)

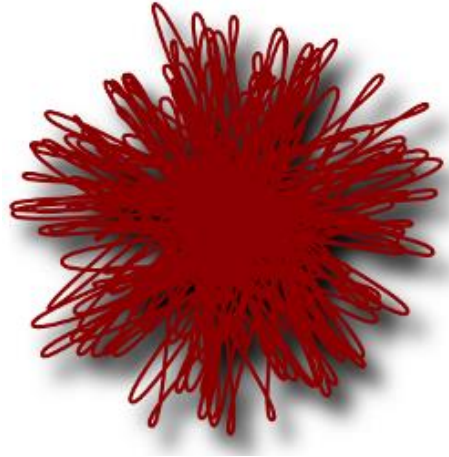
- C, C++, Java, PERL, Python, Android rule conformance checking
- Information flows across applications
- Operating system call flows

Manual review against coding standards

- Web services: SOAP, REST, XML-RPC
- Java applets
- Access control matrices
- Custom network or data protocols.

Conformance Testing Process using SCALe





Available Tools & Training

Tapioca

- a network-layer man-in-the-middle (MITM) proxy VM that is based on UbuFuzz and is preloaded with mitmproxy

BFF/FOE

- a software testing tool that finds defects in applications that run on the Windows, Linux, and Mac OS X platforms

Introductory Fuzzing Course

- Learn how to use CERT's fuzzing tools to crash applications and find exploitable vulnerabilities

SwA for Development

Applying Software Quality Models





TSP is an agile, team-focused process for **high quality** software and systems development.

The TSP strategy improves software engineering from the bottom up.

- Instills engineering discipline in software developers
- Builds high-performance trusted teams

TSP works in practice

Performance Category	Typical TSP Result	Typical Industry Result
Effort estimation error	<10%	>30%
Schedule estimation error	<10%	>30%
Product quality (defects/KLOC)	0.01 to 0.5	1.0 to 7.0



Adding secure design

- Minimize attack surfaces
- Defense in depth for software development

Adding secure coding

- Adopting secure coding practices

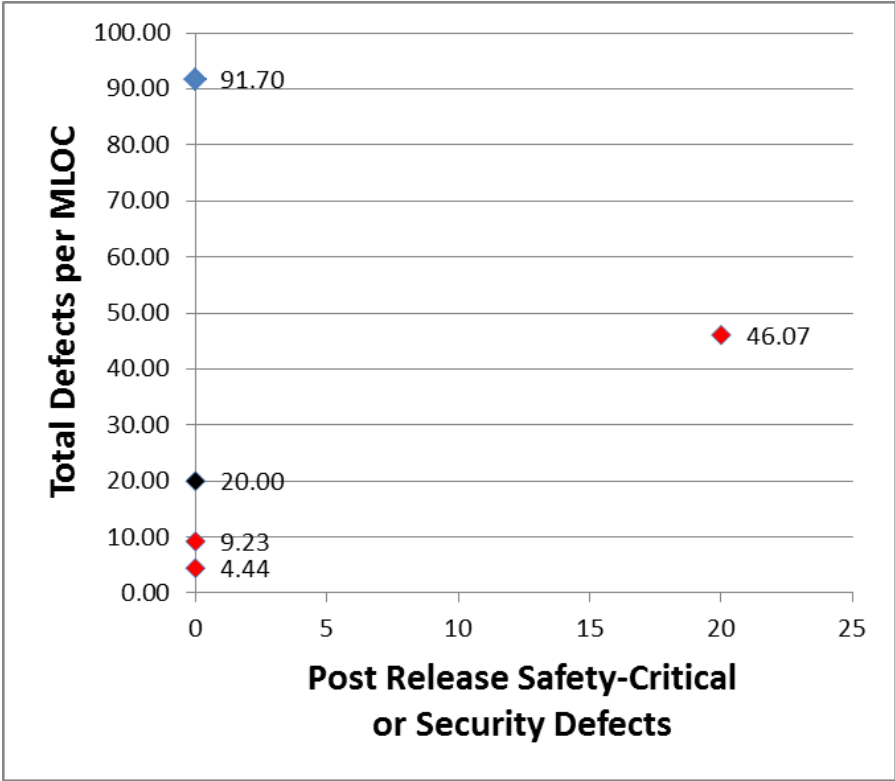
Tooling support for automated conformance checking

Tracking security defects

- Monitoring results of tests with respect to security

Org.	Project	Type	Secure or Safety Critical Defects	Defect Density	Size
D	D1	Safety Critical	20	46.07	2.8 MLOC
D	D2	Safety Critical	0	4.44	.9 MLOC
D	D3	Safety Critical	0	9.23	1.3 MLOC
A	A1	Secure	0	91.70	.6 MLOC
T	T1	Secure	0	20.00	.1 MLOC

Data from five TSP projects with low defect density in system testing reported very low or zero safety critical and security defects in production use.



Woody, Carol et al. *Predicting Software Assurance Using Quality and Reliability Measures*. CMU/SEI-2014-TN-026. Software Engineering Institute, Carnegie Mellon University. 2014. <http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=428589>



Quality tracks defects/faults (engineering and testing)

- **Defect:** non-fulfilment of intended usage requirements (ISO/IEC 9126) [essentially nonconformity to a specified requirement, missing or incorrect requirements]
- **Software fault:** accidental condition that causes a functional unit to fail to perform its required function (IEEE Standard Dictionary of Measures to produce reliable software 982.1, 1988)



Security cares about vulnerabilities (operations)

- **Information security vulnerability:** mistake in software that can be exploited by a hacker to gain access to a system or network (<http://cve.mitre.org/about/terminology.html>)
- **Software vulnerability:** instance of an error in the specification, development, or configuration of software such that its execution can violate a security policy (Shin and Williams, 2010)

Research indicates that **1-5% of defects are vulnerabilities**

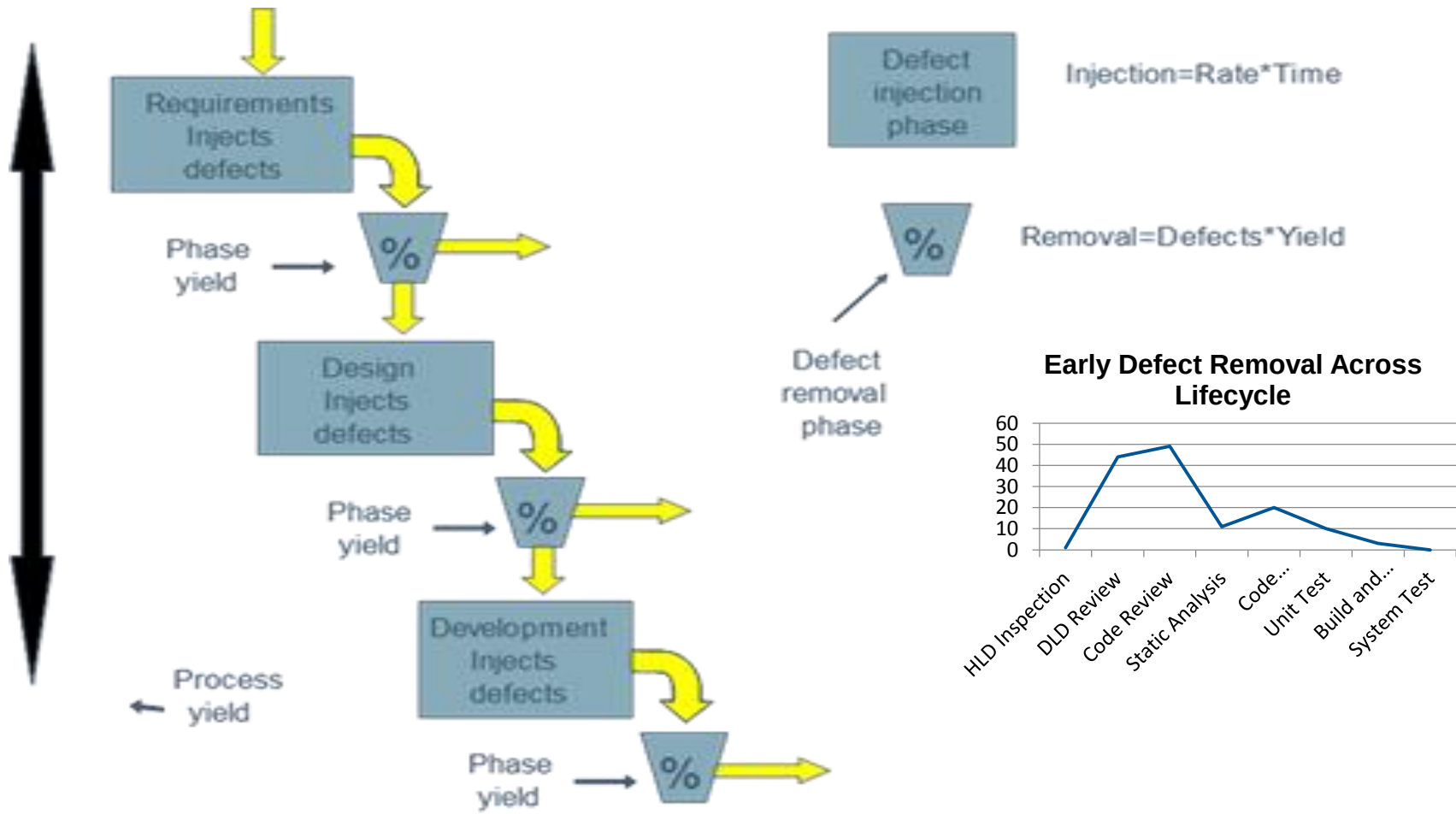
Analysis of defects for five versions of Microsoft Windows operating systems and two versions of Red Hat Linux systems) (Alhazmi, et.al., 2007)

- Win 95 (14.5 MLOC) and Win 98 (18 MLOC) vulnerabilities are 1.00% and 0.84% respectively of identified defects
- Red Hat Linux 6.2 (1.8 MLOC) and 7.1 (6.4 MLOC) vulnerabilities are 5.63% and 4.34% respectively of identified defects.

Tom Longstaff asserted that vulnerabilities might represent 5% of total defects (<http://research.microsoft.com/en-us/um/redmond/events/swsecinstitute/slides/longstaff.pdf>)

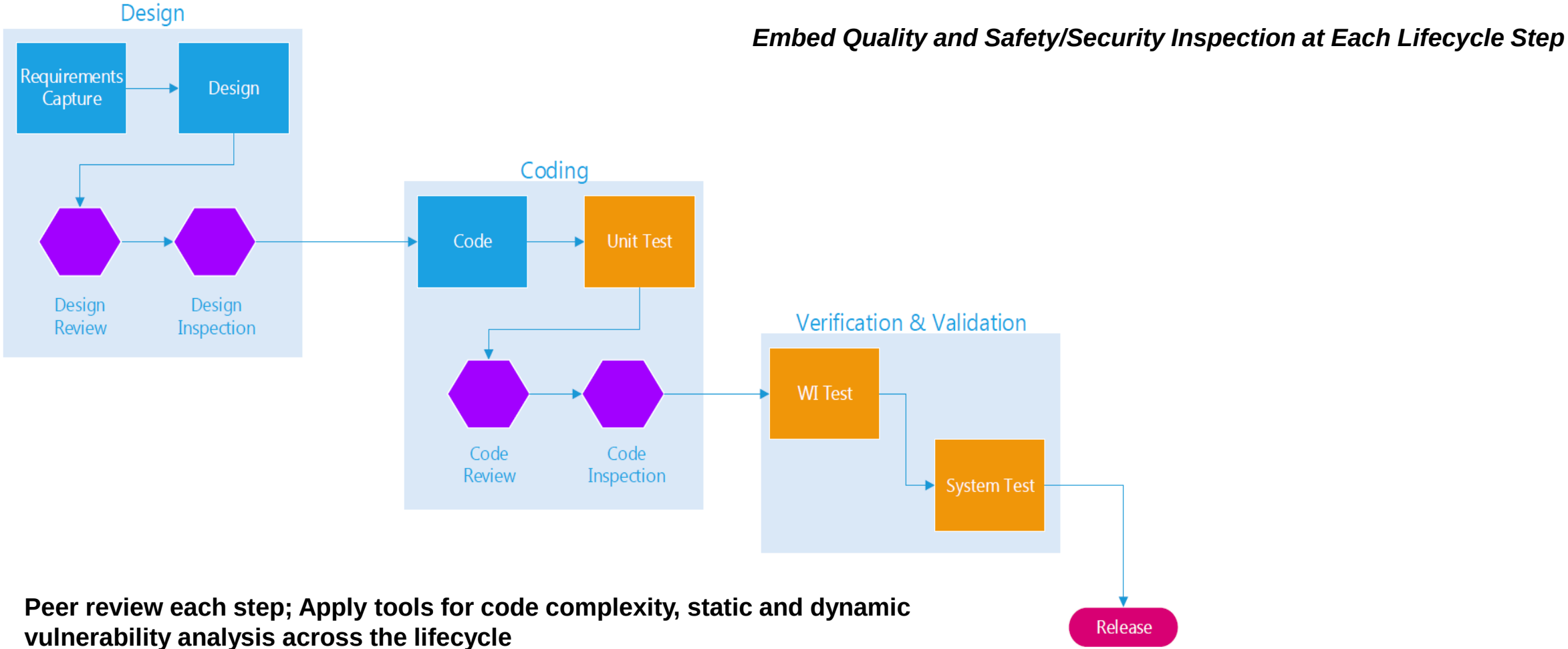
Ross Anderson: “it's reasonable to expect a 35,000,000 line program like Windows 2000 to have 1,000,000 bugs, only 1% of them are security-critical.” (Anderson, 2001)

Quality Focuses on Defect Injection and Removal



Poor quality does predict poor security

Effective quality (early detection) can provide cost-effective security results



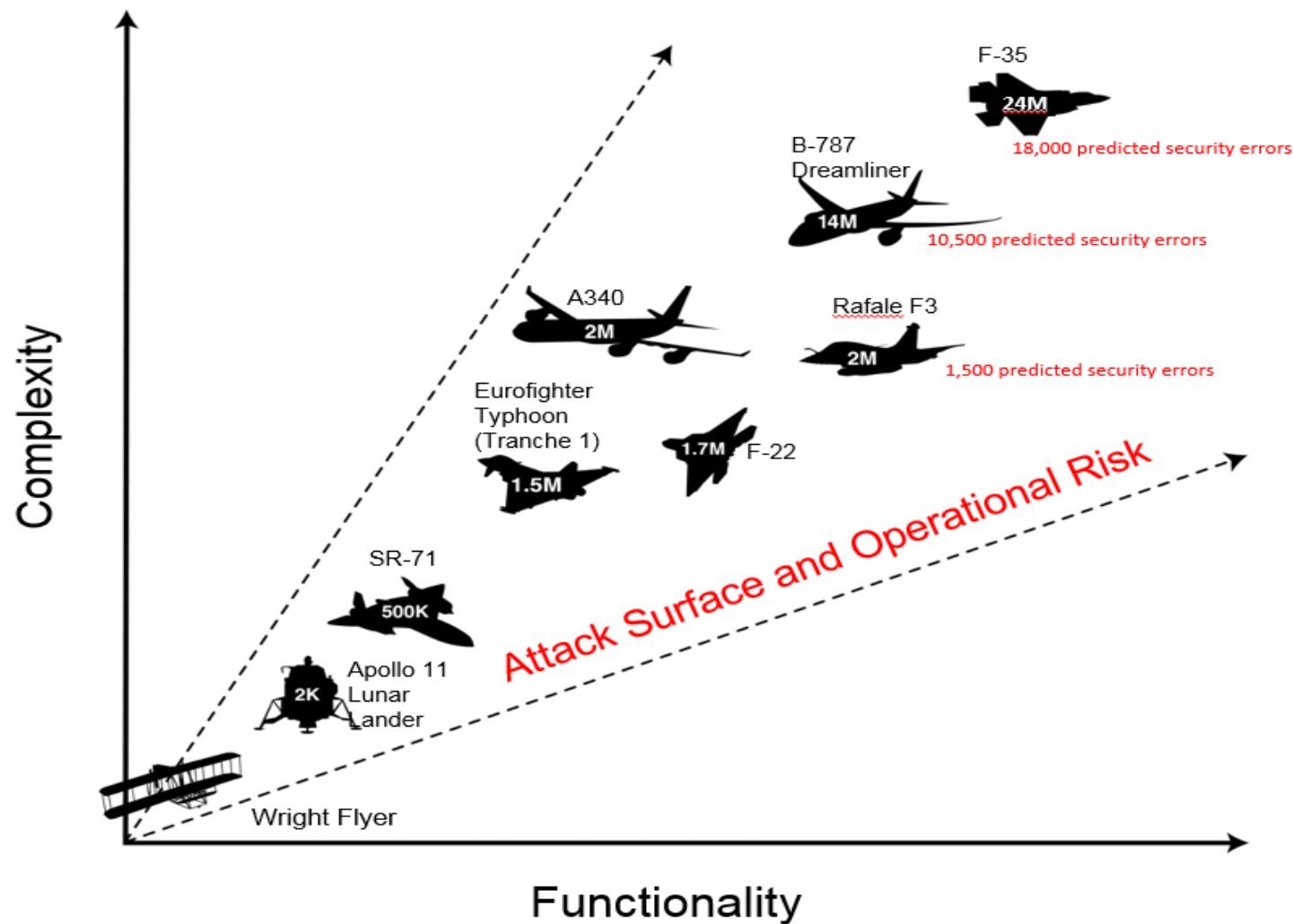
Good quality will ensure proper implementation of specified results

- Effective code checking will identify improper implementations of specifications (11 of SANS Top 25)
- Effective design reviews will identify missing requirements (12 of SANS Top 25)
 - **if** appropriate security results are considered in the development of requirements
 - **if** requirements are effectively translated into detail designs and code specifications to support the required security results

Security requirements must be properly specified

SANS Top 25: SysAdmin, Audit, Network, Security Top 25 Most Dangerous Programming Errors (<http://cwe.mitre.org/top25>)

Increased Software and Complexity -> More Security Issues



Best-in-class code:
2.5 defects per function point and
<600 defects per MLOC

Very good code:
600 to 1,000 defects per MLOC

Average quality code:
4.5 defects per function point and
6000 defects per MLOC.

*Capers Jones,
sqgne.org/presentations
/2011-12/Jones-Sep-2011.pdf*

1-5 % of defects are vulnerabilities.

*Woody, Carol; Ellison, Robert; and
Nichols, William. Predicting Software
Assurance Using Quality and
Reliability Measures. CMU/SEI-2014-
TN-026. Software Engineering
Institute, Carnegie Mellon University.
2014.*

*[http://resources.sei.cmu.edu/library/
asset-view.cfm?AssetID=428589](http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=428589)*

The **Boeing 787 Dreamliner** has 14 MLOC.

- If we assume all of it is **exceptional code**, 8,400 defects and approximately **420 vulnerabilities** remain in the code.
- It is more likely that the code is **average to very good**, which means it could have up to 84,000 defects and **4,200 vulnerabilities**.

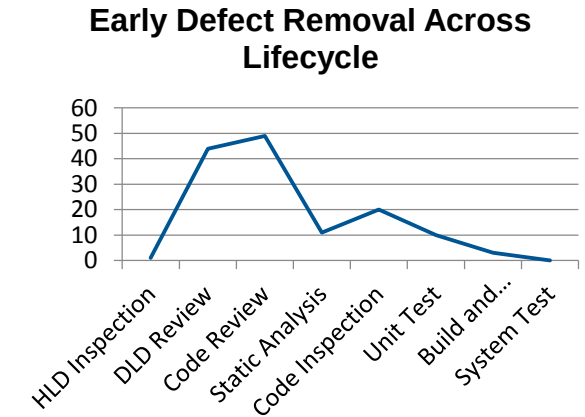
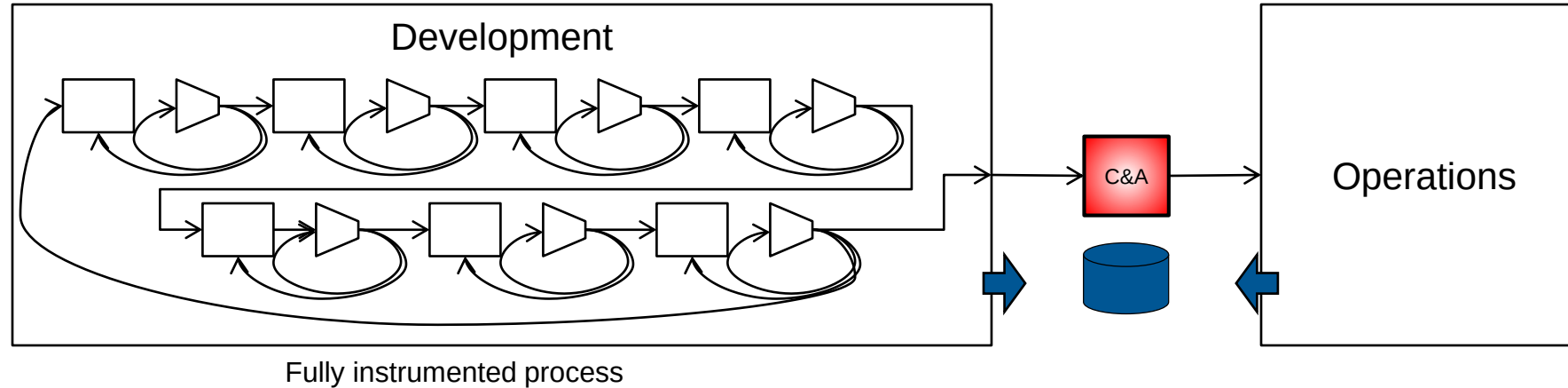
The **F-22** has 1.7 MLOC.

- 1,020–10,200 defects
- 51–510 vulnerabilities

The **F-35 Lightning II** has 24 MLOC.

- 14,400–144,000 defects
- 720–7,200 vulnerabilities

Approach for Predicting Security with Quality



Early lifecycle indications for software assurance risk based on quality processes and quality control

- Fine-grained quality gates and tracking
- Models for predicting phase, increment, release, and operations quality

The Systems Sciences Institute at IBM has reported that “the cost to fix an error found after product release was four to five times as much as one uncovered during design, and up to 100 times more than one identified in the maintenance phase.” [2015]

SwA for Development Summary





Software assurance engineering starts with designing and building security in

- A lifecycle focus is needed to connect security into all aspects of development.
- Security requirements are a critical element of lifecycle software assurance – effective threat modelling is the key to developing good security requirements

Design and code analysis increases confidence that operational security will be effective

- Analyse the design (formal modelling)
- Analyse the code (to ensure the design is properly built and vulnerabilities are reduced)

Software quality is a critical component for software assurance

Software Assurance: Mission Assurance





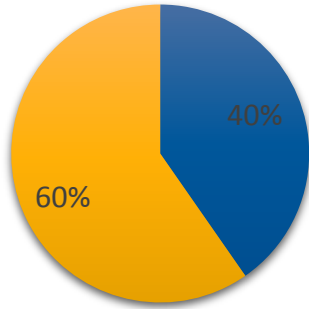
Mission Failure Example – Power Grid 2003

Evaluating Mission Risk

Security Engineering Risk Analysis (SERA)

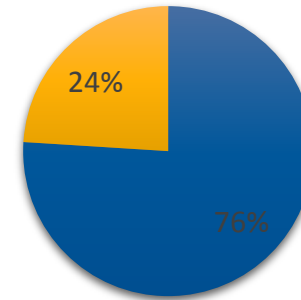
Summary

940 Total CWEs*



■ Design Weakness ■ Other Weakness

Top 25 CWEs (Most Dangerous)



■ Design Weakness ■ Other Weakness

Causes for design weaknesses:

- Poor security requirements
- Limited understanding of the impact of security risk on mission success

Source: <http://cwe.mitre.org/> as of Feb 9, 2014

Mission Assurance

Mission Failure Example

Power Grid 2003



In August, 50 million electricity consumers in Canada and the northeastern U.S. lost power. Multiple failures (handled independently) occurred over a four-hour period.

- Three high-voltage lines went out of service due to tree branches too close to the lines.
- Race condition (software) disabled alarm system that notified operators of line problems.
 - hot backup failed—hardware redundancy only
 - alarm system restart (one of four tightly integrated subsystems) required full control system reboot (> 30 min)
- Operators did not know the alarm system failed; IT did not notify them.
 - operators assumed all was well—no alarms
 - operators did not notice their power failure—automatic power backup

Independent power grid monitoring failed.

System monitoring applications take current state and project future system state. Warning raised when projected state differs significantly from realized state.

- Data errors resulted from the downed lines; IT had to correct errors manually.
- After fixing the data (60 minutes), the IT person forgot to restart the monitor before leaving for lunch.
- Restart failed after lunch as data was out of date.
- Monitor finally restarted about 30 minutes before final failure (insufficient time to analyze discrepancies and respond).

Technical failures required too lengthy a recovery time to be handled independently.

Hardware redundancy will not accommodate software failures.

Critical mission dependencies were not recognized.

- lack of operational ability to identify early warning signs of failure
- independent monitoring not operationally robust

This situation was not triggered by an attacker (but it could have been)

The software did not function as intended

Mission Assurance

Evaluating Mission Risk



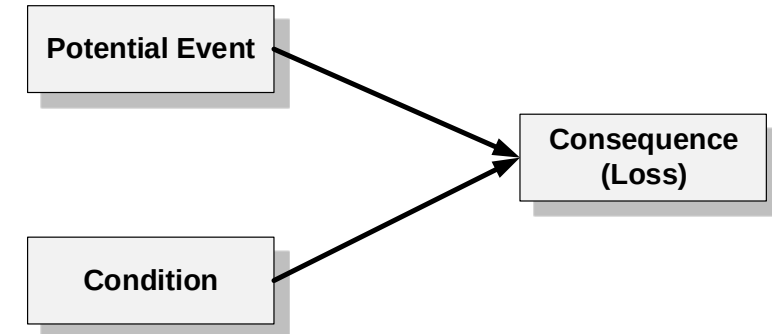
What Is Risk?

The probability of suffering harm or loss

A measure of the likelihood that an event will lead to a loss coupled with the magnitude of the loss

Risk requires the following conditions:¹

- Potential loss
- Likelihood
- Choice



1. Charette, Robert N. *Application Strategies for Risk Analysis*. New York, NY: McGraw-Hill Book Company, 1990.

Probability

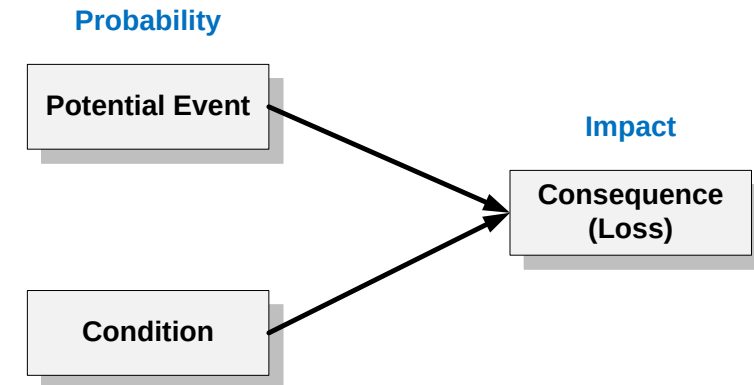
- The likelihood that an event will occur

Impact

- The loss that occurs when a risk is realized

Risk exposure

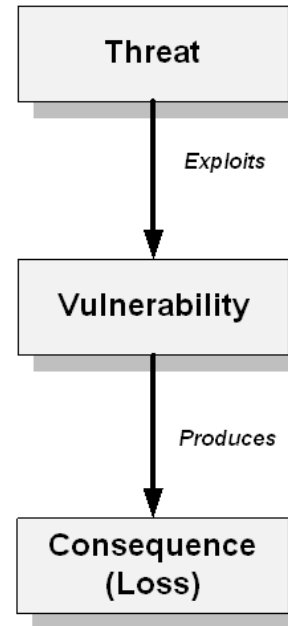
- The magnitude of a risk based on current values of probability and impact



Security risk is a measure of the

1. Likelihood that a threat will exploit a vulnerability to produce an adverse consequence, or loss
2. Magnitude of the loss

Three Components of Security Risk



Threat

- A cyber-based act, occurrence, or event that exploits one or more vulnerabilities and leads to an adverse consequence or loss

Vulnerability

- A weakness in an information system, system security procedures, internal controls or implementation that a threat could exploit to produce an adverse consequence or loss; a current condition that leads to or enables security risk

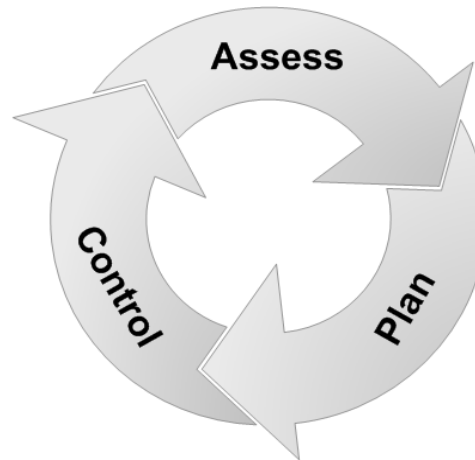
Consequence

- The loss that results when a threat exploits one or more vulnerabilities; the loss is measured in relation to the status quo (i.e., current state)

Systematic approach for minimizing exposure to potential losses.

Risk management provides a disciplined environment for

- Continuously assessing what could go wrong (i.e., assessing risks)
- Determining which risks to address (i.e., setting control priorities)
- Implementing actions to address high-priority risks and bring those risks within tolerance



Assess risk

- Transform the concerns people have into distinct, tangible risks that are explicitly documented and analyzed.

Plan for controlling risk

- Determine an approach for addressing each risk; produce a plan for implementing the approach.

Control risk

- Deal with each risk by implementing its defined control plan and tracking the plan to completion.

Accept—If a risk occurs, its consequences will be tolerated; no proactive action to address the risk will be taken. When a risk is accepted, the rationale for doing so is documented.

Control—Action is taken to handle a risk. Types of control actions include:

- *Transfer*—A risk is shifted to another party (e.g., through insurance or outsourcing).
- *Avoid*—Activities are restructured to eliminate the possibility of a risk occurring.
- *Mitigate*—Actions are implemented in an attempt to reduce or contain a risk.

Recognize and respond

- Monitor the event and take action when it is detected.

Resist

- Implement protection measures to reduce exposure to the event or minimize any consequences that might occur.

Recover

- Return to an acceptable state if the consequences or losses are realized.

Mission Assurance

Security Engineering Risk Analysis (SERA)





What

- A systematic approach for analyzing complex security risks in software-reliant systems and systems of systems across the lifecycle and supply chain

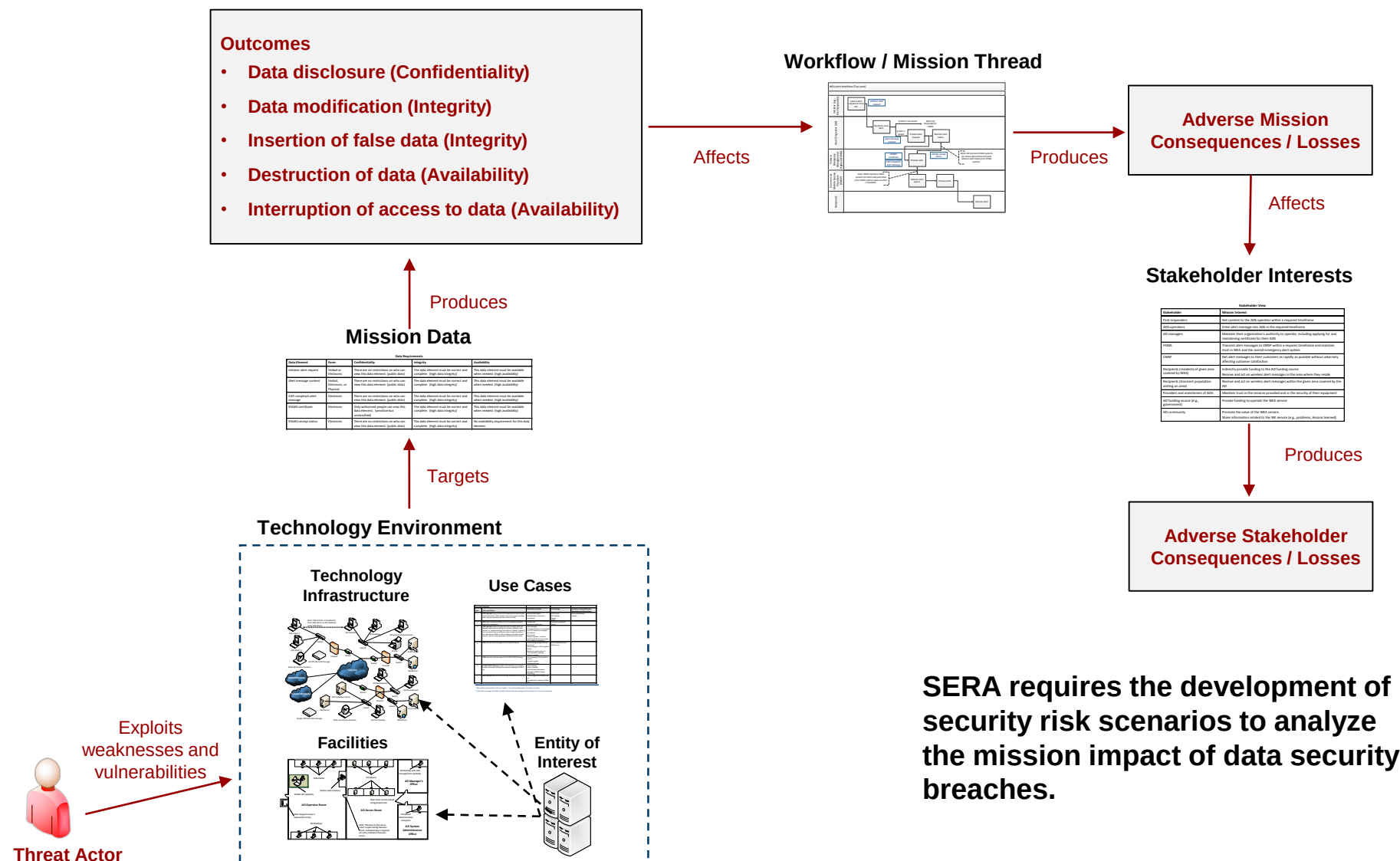
Why

- Build security into software-reliant systems by addressing design weaknesses as early as possible (e.g., requirements, architecture, design)
- Assemble a shared organizational view (business and technical) of cybersecurity risk

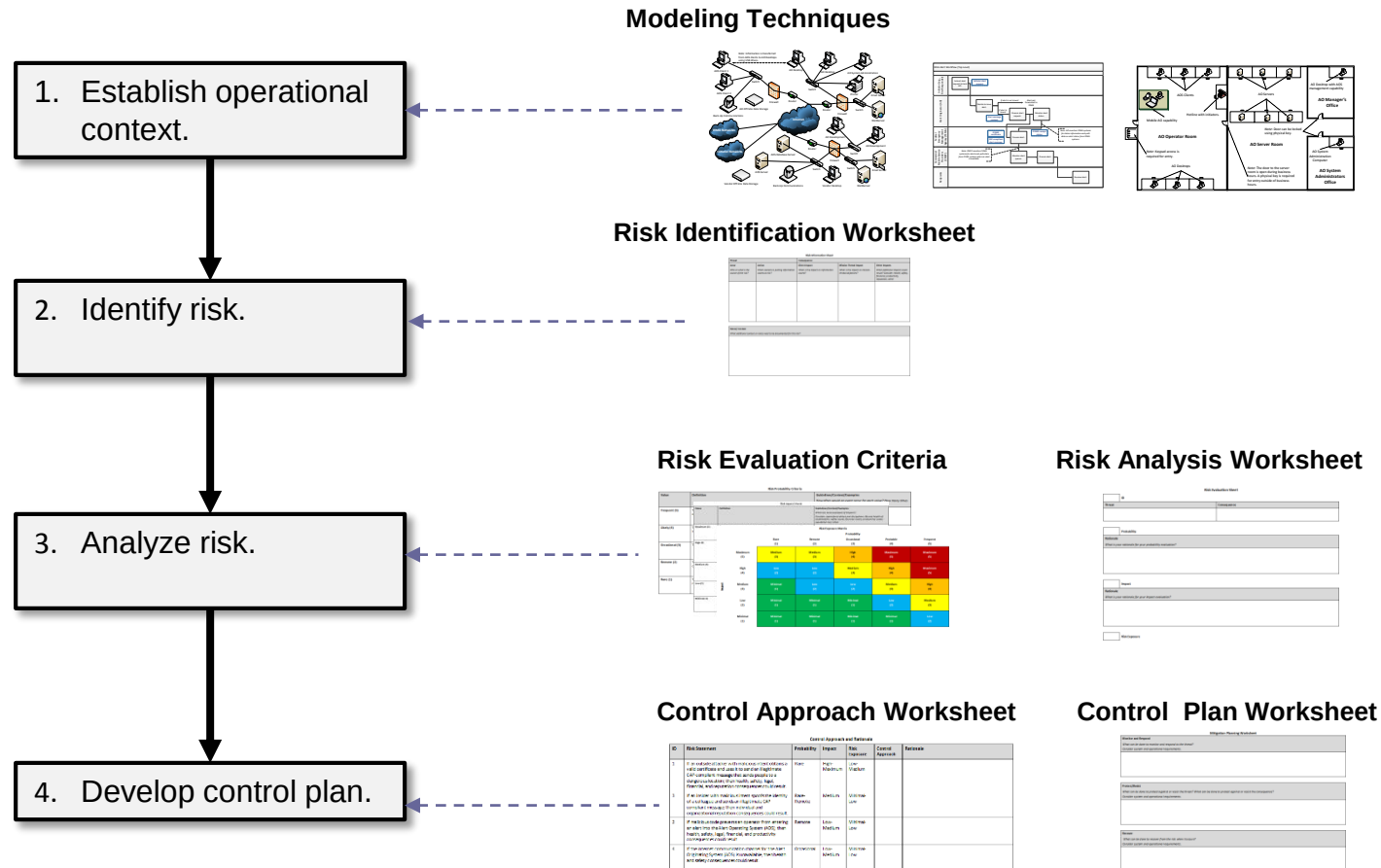
Benefits

- Corrects design weaknesses before a system is deployed
- Reduces residual cybersecurity risk in deployed systems
- Ensures consistency with NIST Risk Management Framework (RMF)

SERA Method: Developing Mission Security Risk Scenarios



SERA Method: *Four Tasks*



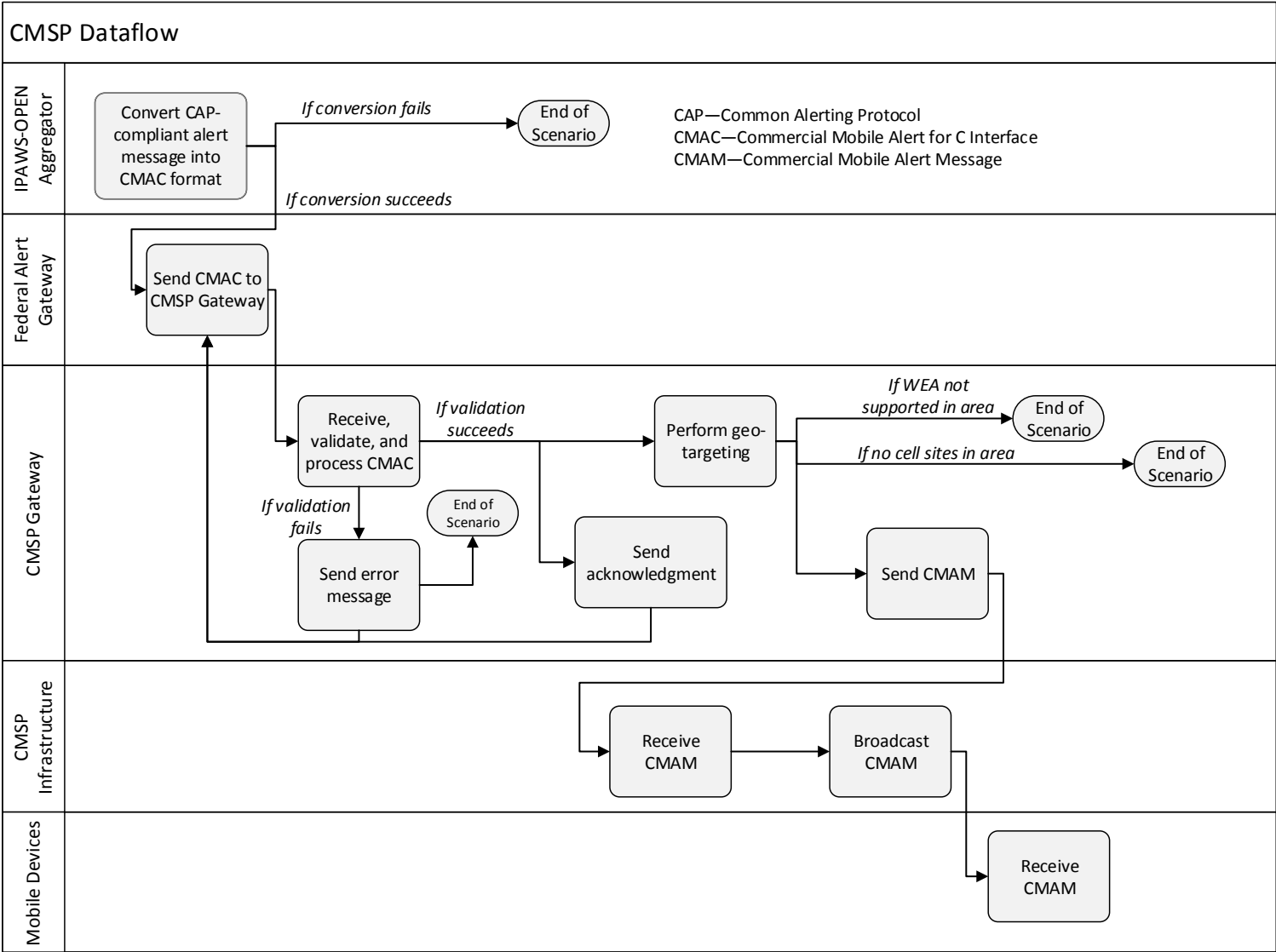
The system of interest (e.g., the software application or system that is being analyzed) is identified.

The operational environment for the system of interest is characterized to establish a baseline of operational performance.

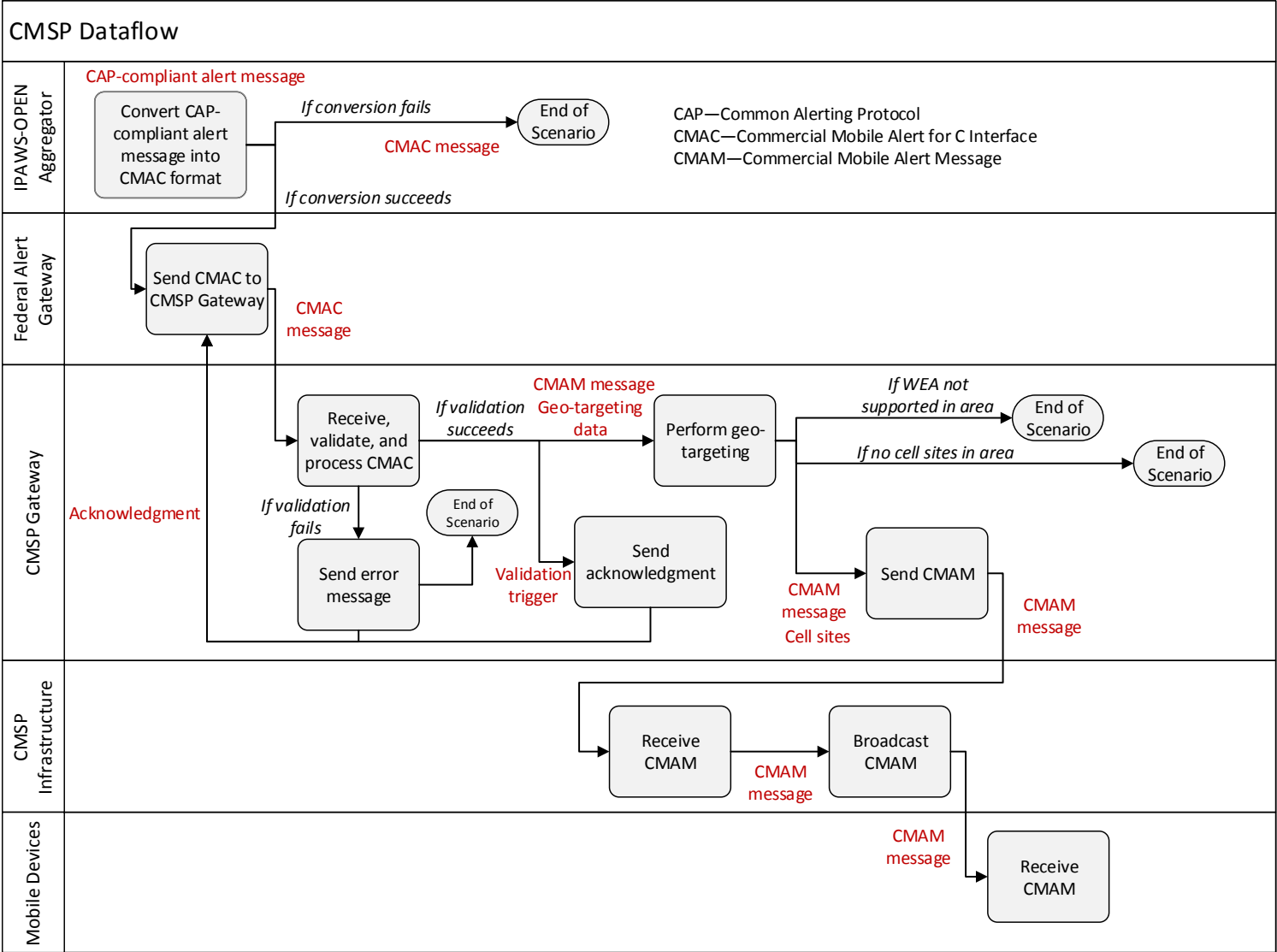
Security risks are analyzed in relation to this baseline.

Steps	
1.1	Determine system of interest.
1.2	Select workflow/mission thread.
1.3	Establish operational views.

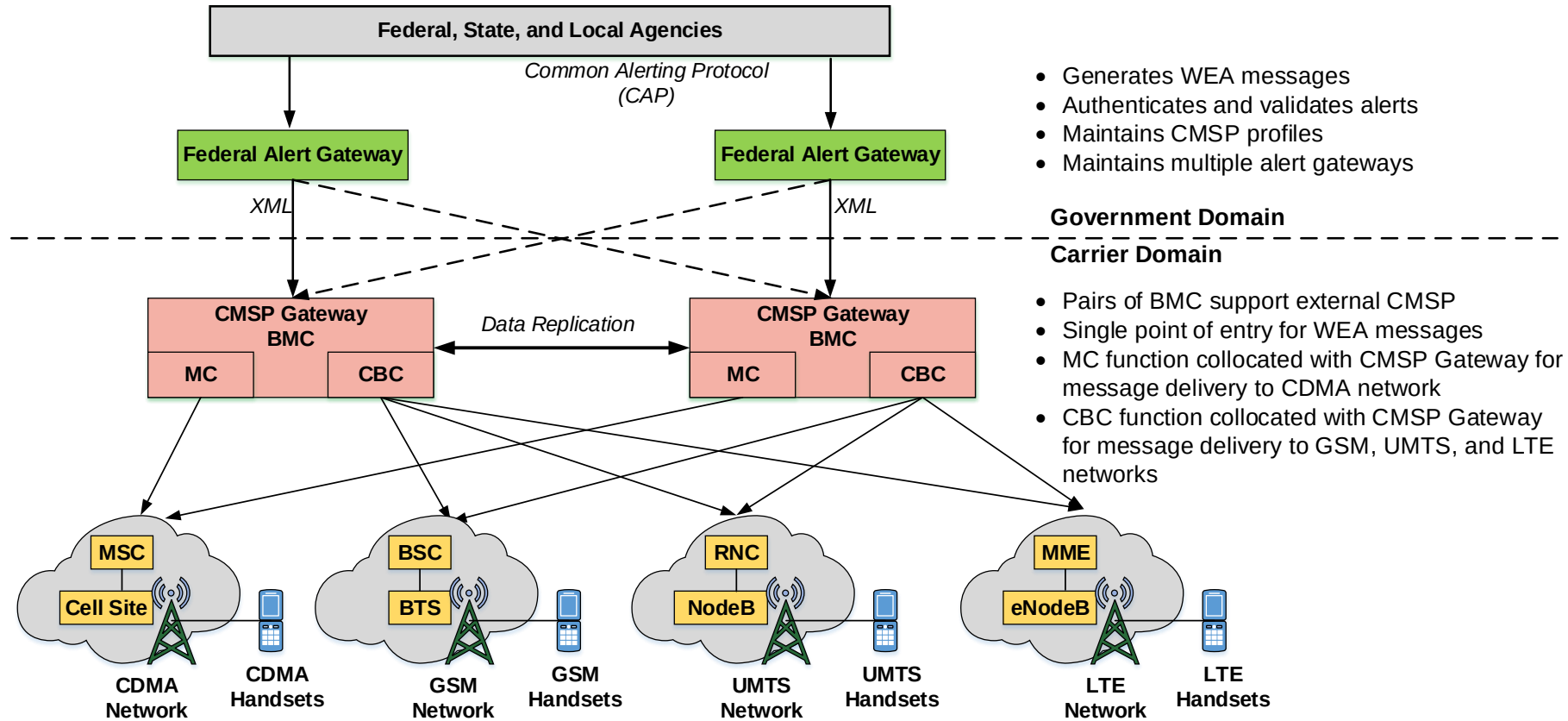
SERA Task 1: Example Workflow



SERA Task 1: Example Dataflow



SERA Task 1: Example Architecture Diagram



Security concerns are transformed into distinct, tangible risk scenarios that can be described and measured.

Steps	
2.1	Identify threat.
2.2	Establish consequence.
2.3	Identify enablers.
2.4	Develop risk scenario.

Threat Components

- Actor – Motive – Goal – Outcome – Means – Threat Complexity

Threat Sequence

- Threat Step – Enabler(s)

Workflow Consequences

- Consequence – Amplifier(s)

Stakeholder Consequences

- Consequence – Amplifier(s)

R1. Insider Sends False Alerts

- **IF** an insider with malicious intent uses the CMSP infrastructure to send nonsense alert messages repeatedly, **THEN** customers could become annoyed with the carrier; the carrier could incur considerable costs to recover from the attack; the carrier's reputation could be tarnished; and public trust in the WEA service could erode.

R2. Inherited Replay Attack

- **IF** the carrier receives emergency alerts from an upstream replay attack on an AO and sends these messages repeatedly to customers in the designated geographic area, **THEN** customers could become annoyed with the carrier; the carrier's reputation could be tarnished; and public trust in the WEA service could erode.

R3. Malicious Code in the Supply Chain

- **IF** malicious code (designed to disseminate alerts as broadly as possible and change the priority of all alerts into Presidential alerts) is inserted into the WEA alerting system by a supply-chain subcontractor, **THEN** customers could become annoyed with the carrier; the carrier could incur considerable costs to recover from the attack; the carrier's reputation could be tarnished; and public trust in the WEA service could erode.

R4. Denial of Service

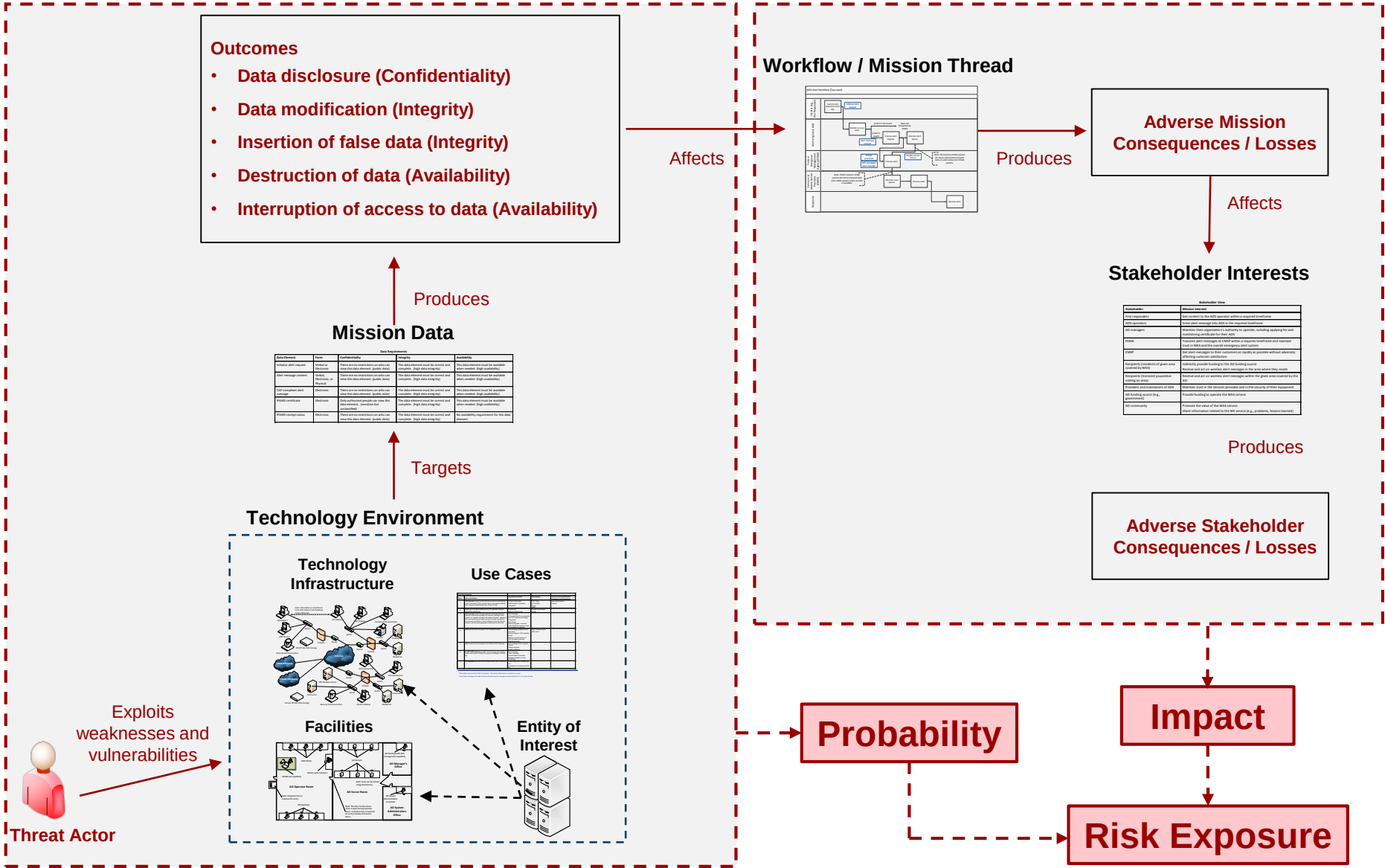
- **IF** an outside actor with malicious intent uses a DoS attack on a carrier's WEA alerting system to prevent the dissemination of an alert about an impending physical terrorist attack, **THEN** people could be unaware of the attack and put in harm's way; the number of injuries and deaths could increase; the carrier could incur considerable costs to recover from the attack; the carrier's reputation could be tarnished; and public trust in the WEA service could erode.

- T1. The insider is upset upon learning that he is not receiving a bonus this year and has been passed over for a promotion.
- T2. The insider begins to behave aggressively and abusively toward his coworkers.
- T3. The insider develops a logic bomb designed to replay a nonsense CMAM message repeatedly.
- T4. The insider uses a colleague's workstation to check-in the modified code with the logic bomb.
- T5. Seven months later, the insider voluntarily leaves the company for a position in another organization.
- T6. Twenty-one days after the insider leaves the carrier, the logic bomb is activated automatically.
- T7. The malicious code causes the carrier's WEA service to send a nonsense WEA alert repeatedly to people across the country.

Each risk is analyzed in relation to predefined criteria.

Steps	
3.1	Establish probability.
3.2	Establish impact.
3.3	Determine risk exposure.

SERA Task 3: Risk Measures



SERA Task 3: *Probability Criteria*

Value	Definition	Guidelines/Context/Examples
Frequent (5)	The scenario occurs on numerous occasions or in quick succession. It tends to occur quite often or at close intervals.	$\geq$ one time per month (≥ 12 / year)
Likely (4)	The scenario occurs on multiple occasions. It tends to occur reasonably often, but not in quick succession or at close intervals.	
Occasional (3)	The scenario occurs from time to time. It tends to occur "once in a while."	$\sim$ one time per 6 months (~ 2 / year)
Remote (2)	The scenario can occur, but it is not likely to occur. It has "an outside chance" of occurring.	
Rare (1)	The scenario infrequently occurs and is considered to be uncommon or unusual. It is not frequently experienced.	$\leq$ one time every 3 years ($\leq .33$ / year)

SERA Task 3: *Impact Criteria*

Value	Definition
Maximum (5)	The impact on the organization is severe. Damages are extreme in nature. Mission failure has occurred. Stakeholders will lose confidence in the organization and its leadership. The organization either will not be able to recover from the situation, or recovery will require an extremely large investment of capital and resources. Either way, the future viability of the organization is in doubt.
High (4)	The impact on the organization is large. Significant problems and disruptions are experienced by the organization. As a result, the organization will not be able to achieve its current mission without a major re-planning effort. Stakeholders will lose some degree of confidence in the organization and its leadership. The organization will need to reach out to stakeholders aggressively to rebuild confidence. The organization should be able to recover from the situation in the long run. Recovery will require a significant investment of organizational capital and resources.
Medium (3)	The impact on the organization is moderate. Several problems and disruptions are experienced by the organization. As a result, the organization will not be able to achieve its current mission without some adjustments to its plans. The organization will need to work with stakeholders to ensure their continued support. Over time, the organization will be able to recover from the situation. Recovery will require a moderate investment of organizational capital and resources.
Low (2)	The impact on the organization is relatively small, but noticeable. Minor problems and disruptions are experienced by the organization. The organization will be able to recover from the situation and meet its mission. Recovery will require a small investment of organizational capital and resources.
Minimal (1)	The impact on the organization is negligible. Any damages can be accepted by the organization without affecting operations or the mission being pursued. No stakeholders will be affected. Any costs incurred by the organization will be incidental.

Current Probability: Remote

Risk Exposure Matrix

Current Impact:
Medium

Impact

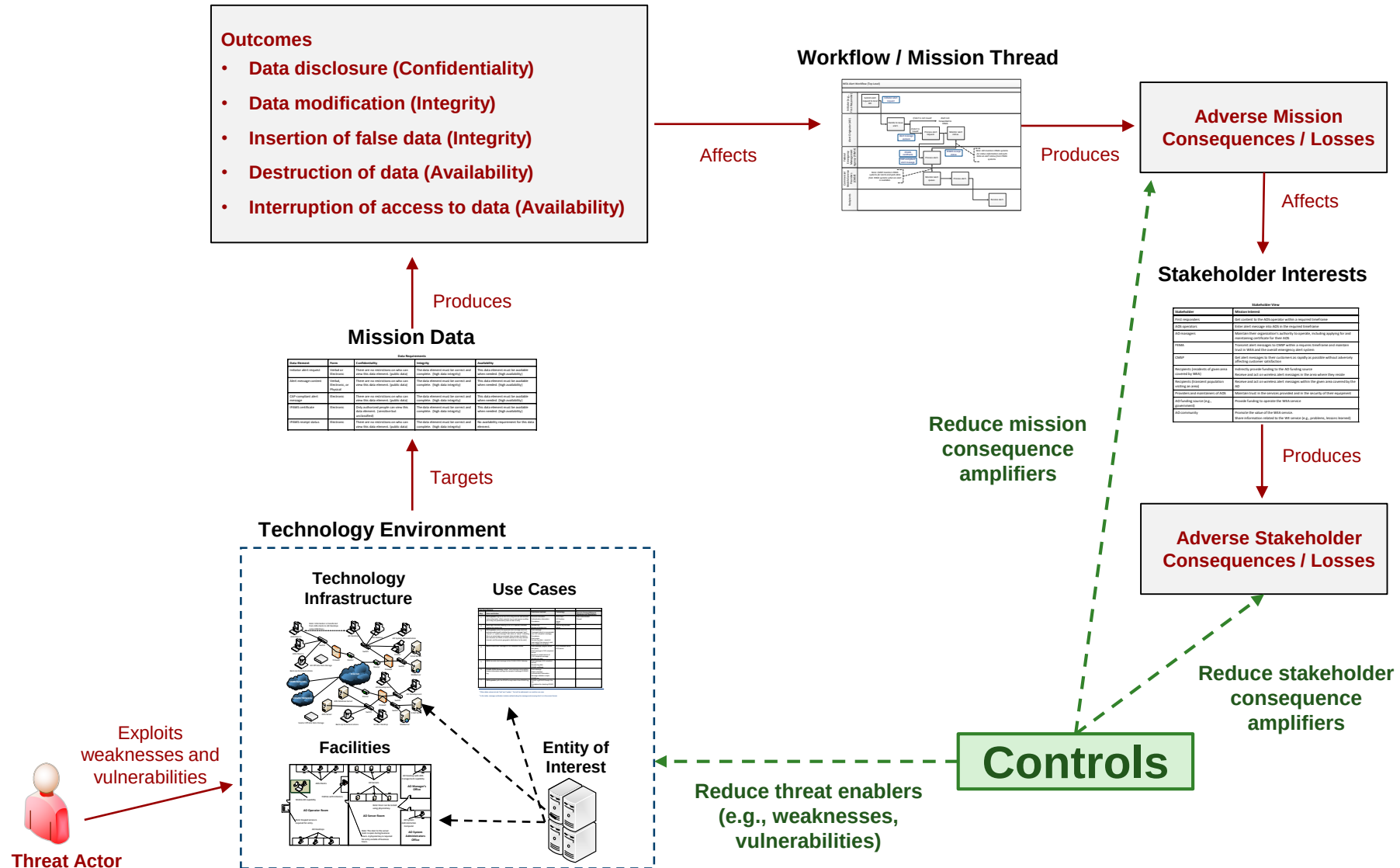
		Probability				
		Rare (1)	Remote (2)	Occasional (3)	Probable (4)	Frequent (5)
Impact	Maximum (5)	Medium (3)	Medium (3)	High (4)	Maximum (5)	Maximum (5)
	High (4)	Low (2)	Low (2)	Medium (3)	High (4)	Maximum (5)
	Medium (3)	Minimal (1)	Low (2)	Low (2)	Medium (3)	High (4)
	Low (2)	Minimal (1)	Minimal (1)	Minimal (1)	Low (2)	Medium (3)
	Minimal (1)	Minimal (1)	Minimal (1)	Minimal (1)	Minimal (1)	Low (2)

A strategy for controlling each risk is determined.
Control plans are developed and documented for all security risks that are not accepted.

Steps	
4.1	Prioritize risks.
4.2	Select control approach.
4.3	Establish control actions.

SERA Task 4: *Prioritized Risk Spreadsheet*

ID	Risk Statement	Imp	Prob	RE
R4	Denial of Service: IF an outside actor with malicious intent uses a DoS attack on a carrier's WEA alerting system to prevent the dissemination of an alert about an impending physical terrorist attack, THEN people could be unaware of the attack and put in harm's way; the number of injuries and deaths could increase; the carrier could incur considerable costs to recover from the attack; the carrier's reputation could be tarnished; and public trust in the WEA service could erode.	Max	Rare	Med
R1	Insider Sends False Alerts: IF an insider with malicious intent uses the CMSP infrastructure to send nonsense alert messages repeatedly, THEN customers could become annoyed with the carrier; the carrier could incur considerable costs to recover from the attack; the carrier's reputation could be tarnished; and public trust in the WEA service could erode.	Med	Remote	Low
R2	Inherited Replay Attack: IF the carrier receives emergency alerts from an upstream replay attack on an AO and sends these messages repeatedly to customers in the designated geographic area, THEN customers could become annoyed with the carrier; the carrier's reputation could be tarnished; and public trust in the WEA service could erode.	Med	Remote	Low
R3	Malicious Code in the Supply Chain: IF malicious code (designed to disseminate alerts as broadly as possible and change the priority of all alerts into Presidential alerts) is inserted into the WEA alerting system by a supply-chain subcontractor, THEN customers could become annoyed with the carrier; the carrier could incur considerable costs to recover from the attack; the carrier's reputation could be tarnished; and public trust in the WEA service could erode.	Med	Rare	Min



SERA Task 4: Control-to-Risk Mapping (Excerpt)

Category	Control	R1	R2	R3	R4
Human Resources	The carrier's managers are trained to provide constructive feedback on performance issues.	X			
	The carrier's managers recognize inappropriate behavior when it occurs and respond appropriately.	X			
	The carrier performs targeted monitoring of individuals with suspected behavioral issues and responds appropriately.	X			
Physical Security	The carrier implements physical access controls for workstations and workspaces.	X			
System Architecture	Security controls are implemented in systems and network devices based on cybersecurity risk.				X
	The carrier's WEA alerting system has a backup capability that uses a separate communication channel.				X
Technical Monitoring	The carrier monitors messages for suspicious content (e.g., illegitimate messages, duplicate messages) and responds appropriately.	X	X	X	
	The carrier monitors its network for abnormal activity (e.g., abnormal traffic patterns, spikes in traffic) and responds appropriately.	X	X	X	X
Contracting	All contracts with third parties specify security standards that must be met.			X	X

The CMSP Cybersecurity Guidelines comprise 35 high-priority security controls that address the four WEA risk scenarios included in this study

Controls were identified in the following areas:

- Human Resources
- Training
- Contracting
- Physical Security
- Change Management
- Access Control
- Information Management
- Vulnerability Management
- System Architecture
- System Configuration
- Code Analysis
- Technical Monitoring
- Independent Reviews
- Incident Response
- Disaster Recovery

Mission Assurance Summary





Understanding mission impact is critical to identifying and prioritizing security risks

SERA provides a methodology for identifying and analyzing security mission risk

- Supports failure analysis and mission impact of interacting systems and components
- Identifies gaps in system requirements with the analysis and evaluation of failure potential and mission assurance

Technical note describing SERA (other references on Slide 251) <http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=427321>

Software Assurance: SwA for Acquisition





Risk in the Software Supply Chain

Software Supply Chain Risk Management (SCRM)

External Dependencies - Services

SCRM Standards

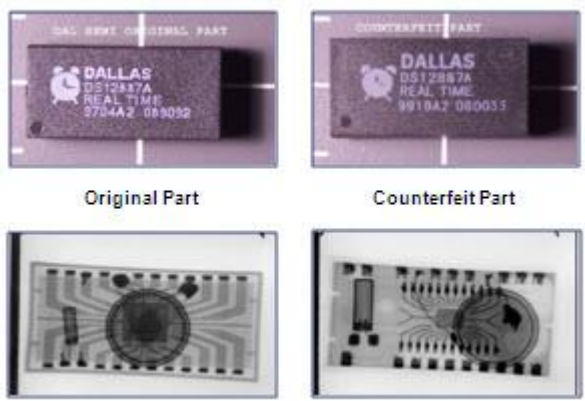
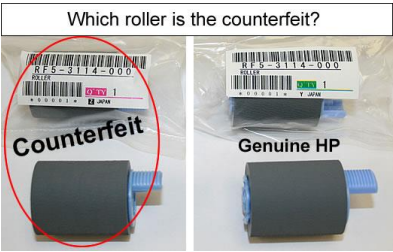
Summary

SwA for Acquisition

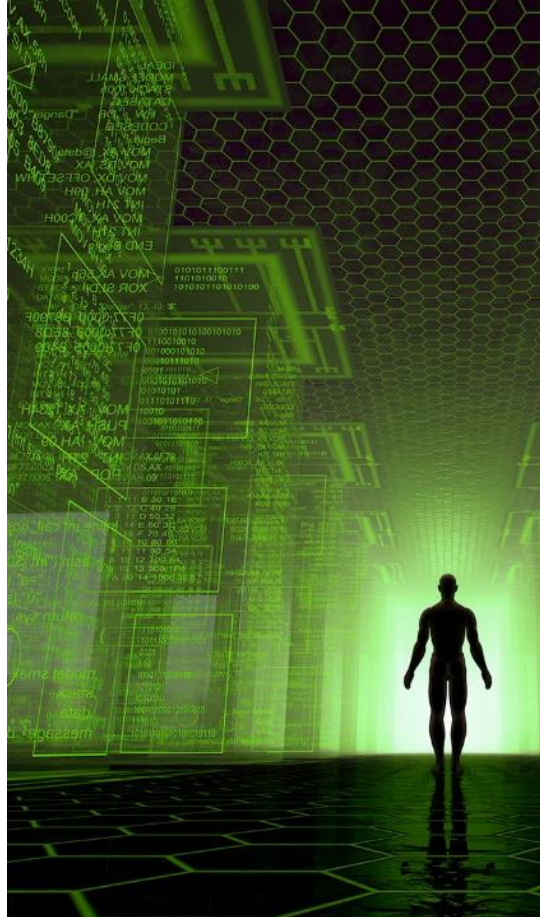
Risk in the Software Supply Chain



Conventional View of Supply Chain Risk



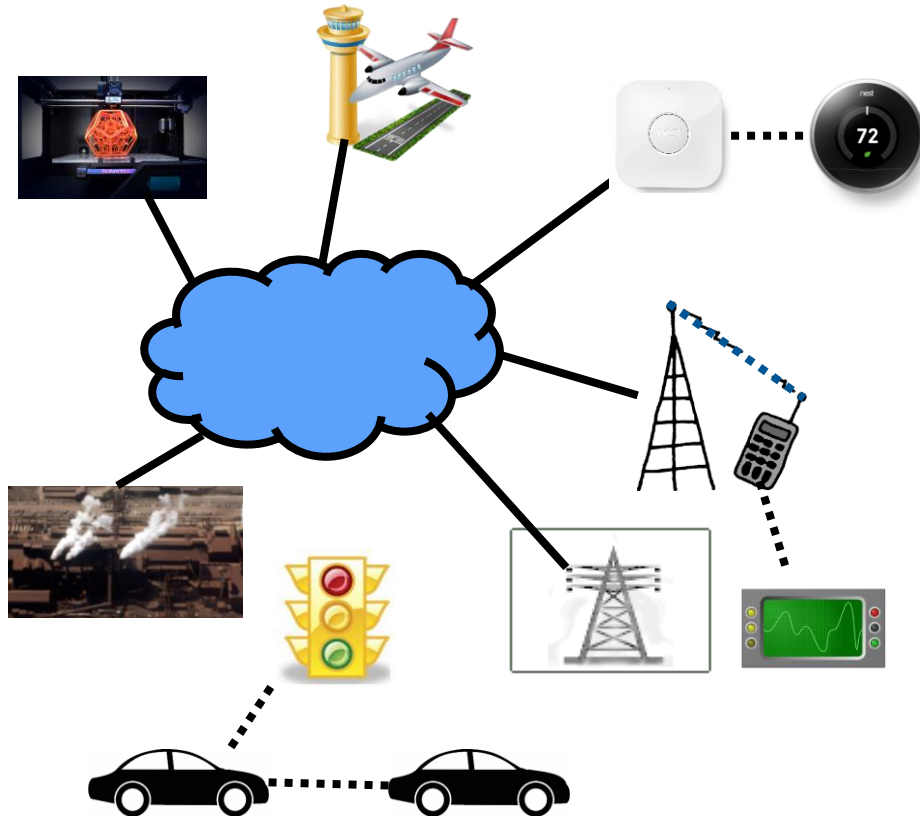
Sources: <http://www.muscatdaily.com/Archive/Oman/Fake-car-parts-contribute-to-rise-in-road-accidents-Experts;>
[http://www.andovercg.com/services/cisco-counterfeit-wic-1dsu-t1.shtml;](http://www.andovercg.com/services/cisco-counterfeit-wic-1dsu-t1.shtml) <http://unites-systems.com/l.php?id=191>



IT moving from specialized hardware to software, virtualized as

- Servers: virtual CPUs
- Storage: SANs
- Switches: Soft switches
- Networks: Software defined networks
- Communications: Software defined radios

Software is the New Hardware – Cyber Physical



- Cellular
 - Main processor
 - Graphics processor
 - Base band processor (SDR)
 - Secure element (SIM)
- Automotive
 - Autonomous vehicles
 - Vehicle to infrastructure (V2I)
 - Vehicle to vehicle (V2V)
- Industrial and home automation
 - 3D printing (additive manufacturing)
 - Autonomous robots
 - Interconnected SCADA
- Aviation
 - Next Gen air traffic control
 - Fly by wire
- Smart grid
 - Smart electric meters
 - Smart metering infrastructure
- Embedded medical devices



Custom development – context:

- Software was limited
 - Size
 - Function
 - Audience
- Each organization employed developers
- Each organization created their own software

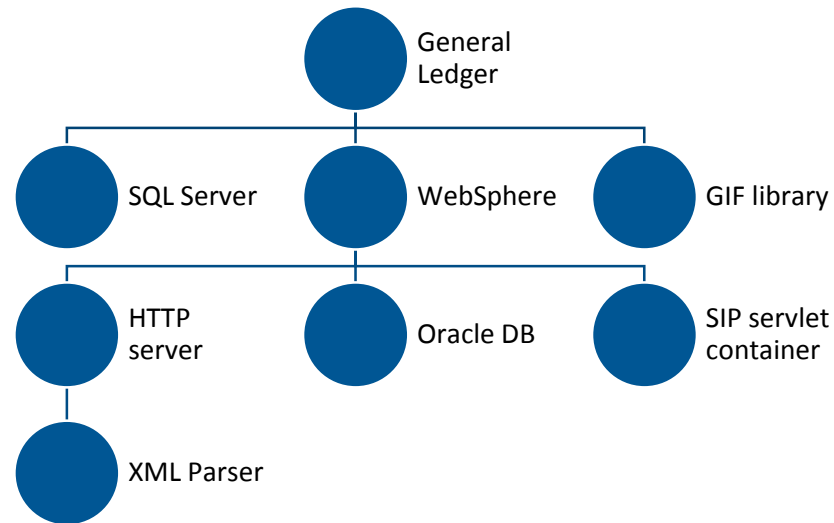
**Supply chain:
practically none**



Shared development – Independent Software Vendors (ISVs) provide Commercial Off the Shelf Software (COTS) – context:

- Function largely understood
 - Automating existing processes
- Grown beyond ability for using organization to develop economically
- Outside of core competitiveness by acquirers

**Supply chain:
software supplier**



Note: hypothetical application composition

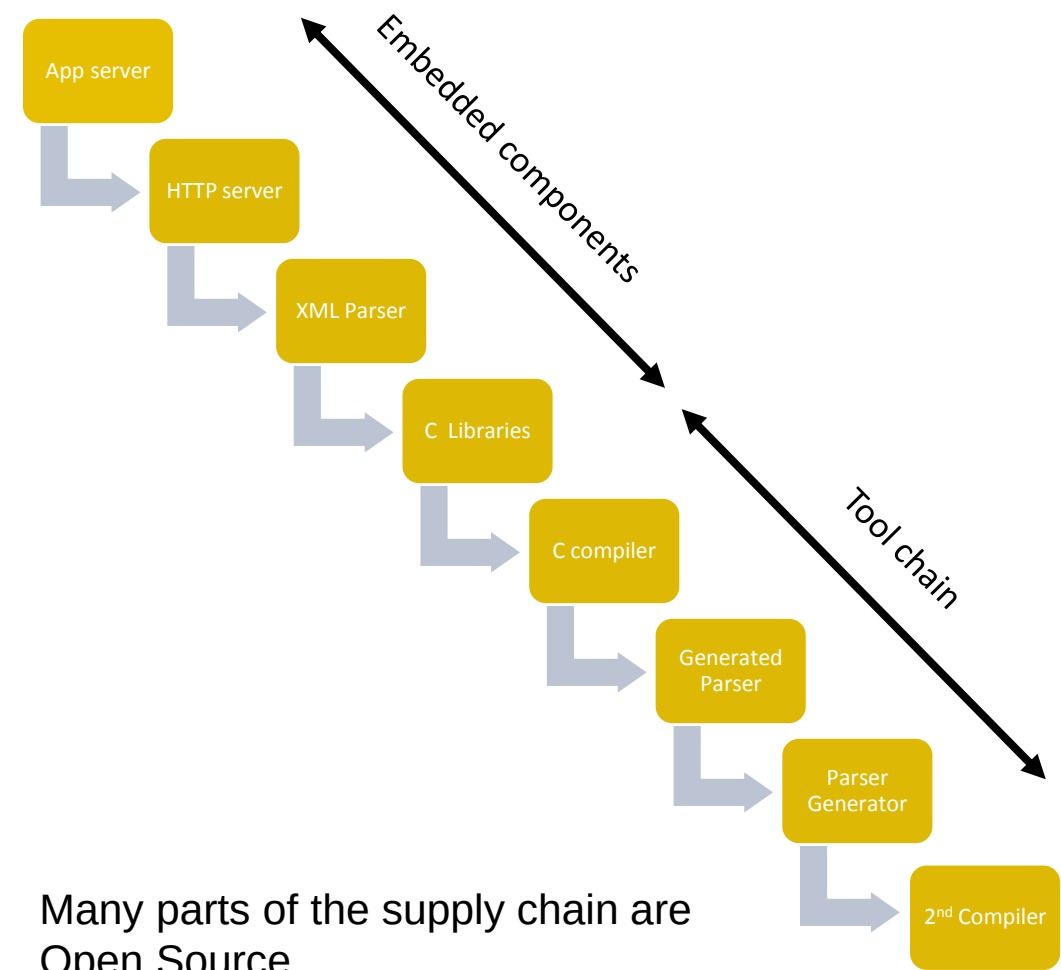
Development is now assembly

using collective development

- Too large for single organization
- Too much specialization
- Too little value in individual components

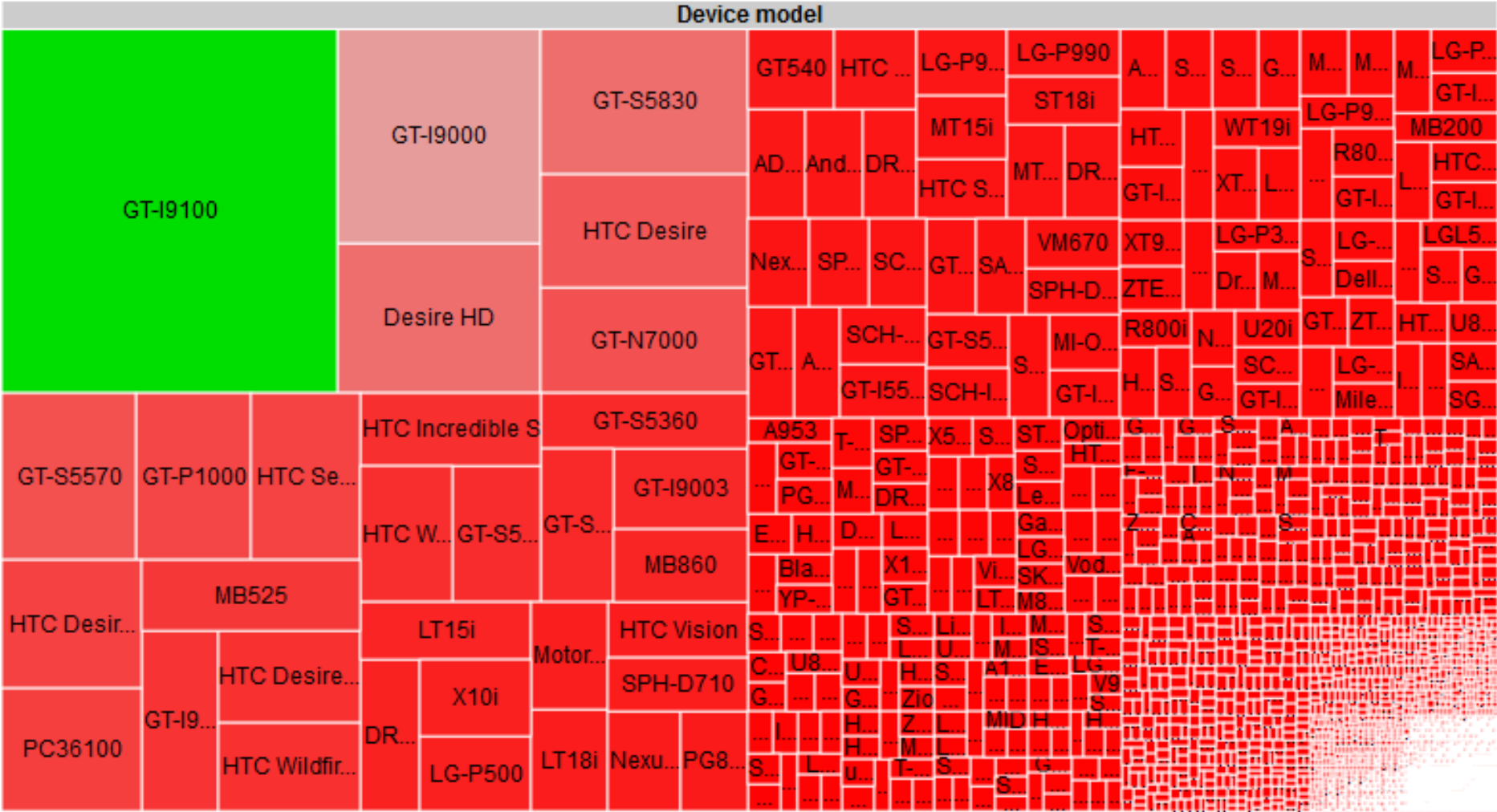
Supply chain: complex

Supply Chains have a Long Path



Many parts of the supply chain are Open Source

Versions of Android Illustrate Open Source Fragmentation



Source: <http://opensignal.com/reports/fragmentation.php>

Open Source is Not Secure



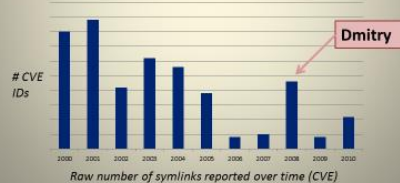
Heartbleed and Shellshock were found by exploitation

Other open source software illustrates vulnerabilities from cursory inspection

Grep-and-Gripe: Revenge of the Symlinks

```
grep -A5 -B5 /tmp/ $PROGRAM
```

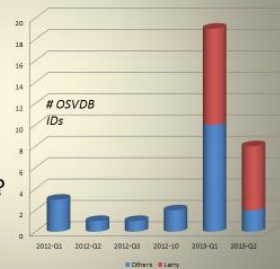
- Dmitry E. Oboukhov, August 2008
- Run against Debian packages
- This kind of thing really hurts pie charts of different vulnerability types



Grep-and-Gripe 2: Larry Cashdollar*

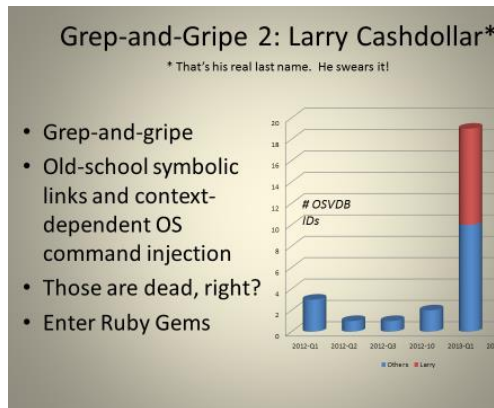
* That's his real last name. He swears it!

- Grep-and-gripe
- Old-school symbolic links and context-dependent OS command injection
- Those are dead, right?
- Enter Ruby Gems



Sources: Steve Christey (MITRE) & Brian Martin (OSF), Buying Into the Bias: Why Vulnerability Statistics Suck, <https://media.blackhat.com/us-13/US-13-Martin-Buying-Into-The-Bias-Why-Vulnerability-Statistics-Suck-Slides.pdf>; Sonatype, Sonatype Open Source Development and Application Security Survey; Sonatype, 2016 State of the Software Supply Chain; Aspect Software “The Unfortunate Reality of Insecure Libraries,” March 2012

Open Source is Not Secure



1.8 billion vulnerable open source components downloaded in 2015

26% of the most common open source components have high risk vulnerabilities

On average, applications have 22.5 open source vulnerabilities

Heartbleed and Shellshock were found by exploitation

Other open source software illustrates vulnerabilities from cursory inspection

Sources: Steve Christey (MITRE) & Brian Martin (OSF), Buying Into the Bias: Why Vulnerability Statistics Suck, <https://media.blackhat.com/us-13/US-13-Martin-Buying-Into-The-Bias-Why-Vulnerability-Statistics-Suck-Slides.pdf>; Sonatype, Sonatype Open Source Development and Application Security Survey; Sonatype, 2016 State of the Software Supply Chain; Aspect Software "The Unfortunate Reality of Insecure Libraries," March 2012

SwA for Acquisition

Software Supply Chain Risk Management



Acquirer

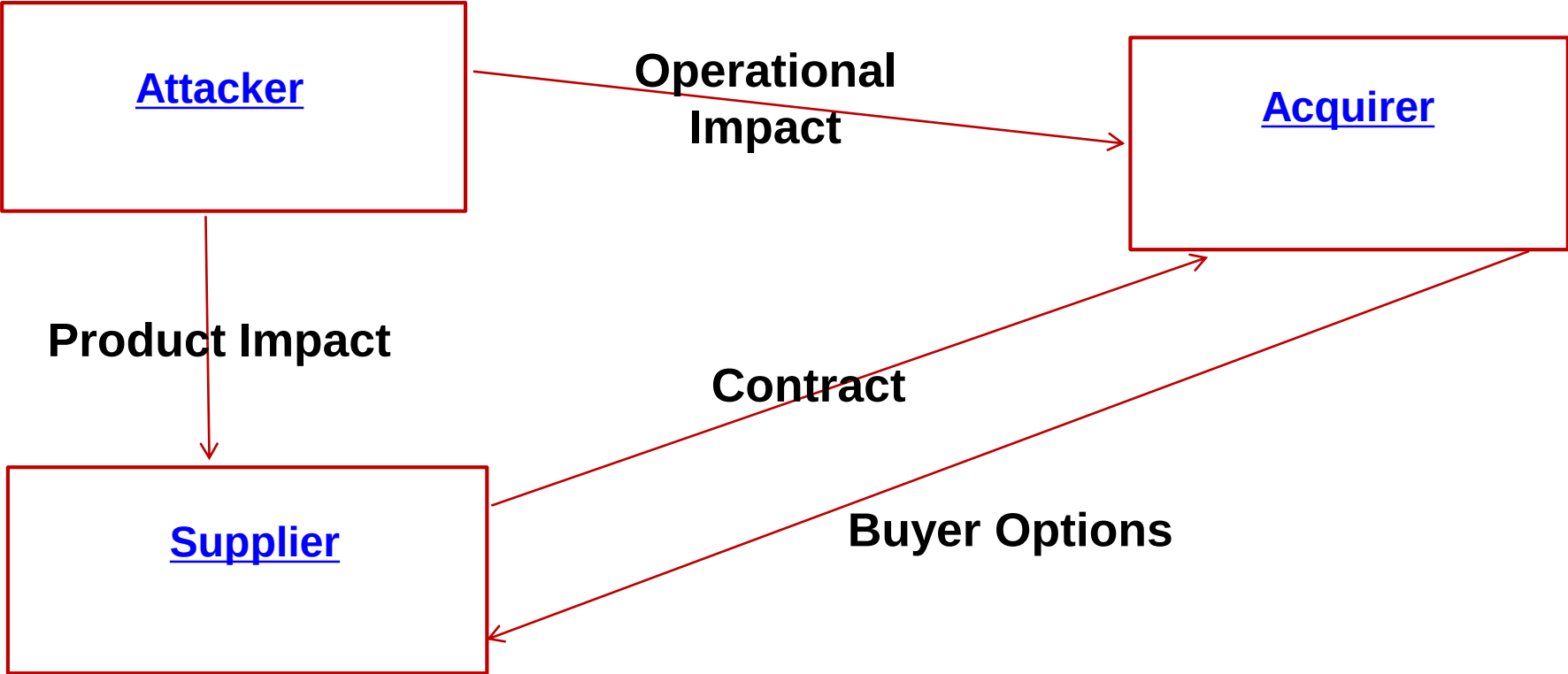
- Product acquisition
- System/Service acquisition

Supplier

- Prime contractor
- Sub-contractors

Attacker

- Target of opportunity
- Focused targeting





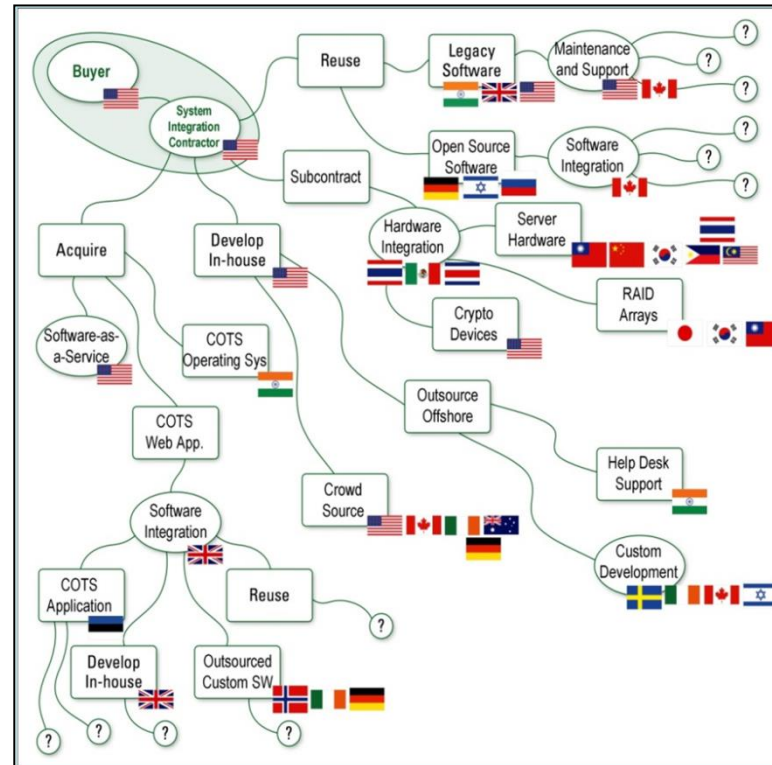
Who makes supply chain decisions?

Who owns the software assurance risk?

Where is the impact of an attack realized?

Potential SCRM Participants

- Information security
- Information assurance
- System and software engineering
- Supply chain logistics
- Process improvement
- Policy
- Legal
- Quality



What

- Assess gaps in acquisition and development supply chain practices that can lead to cybersecurity risk

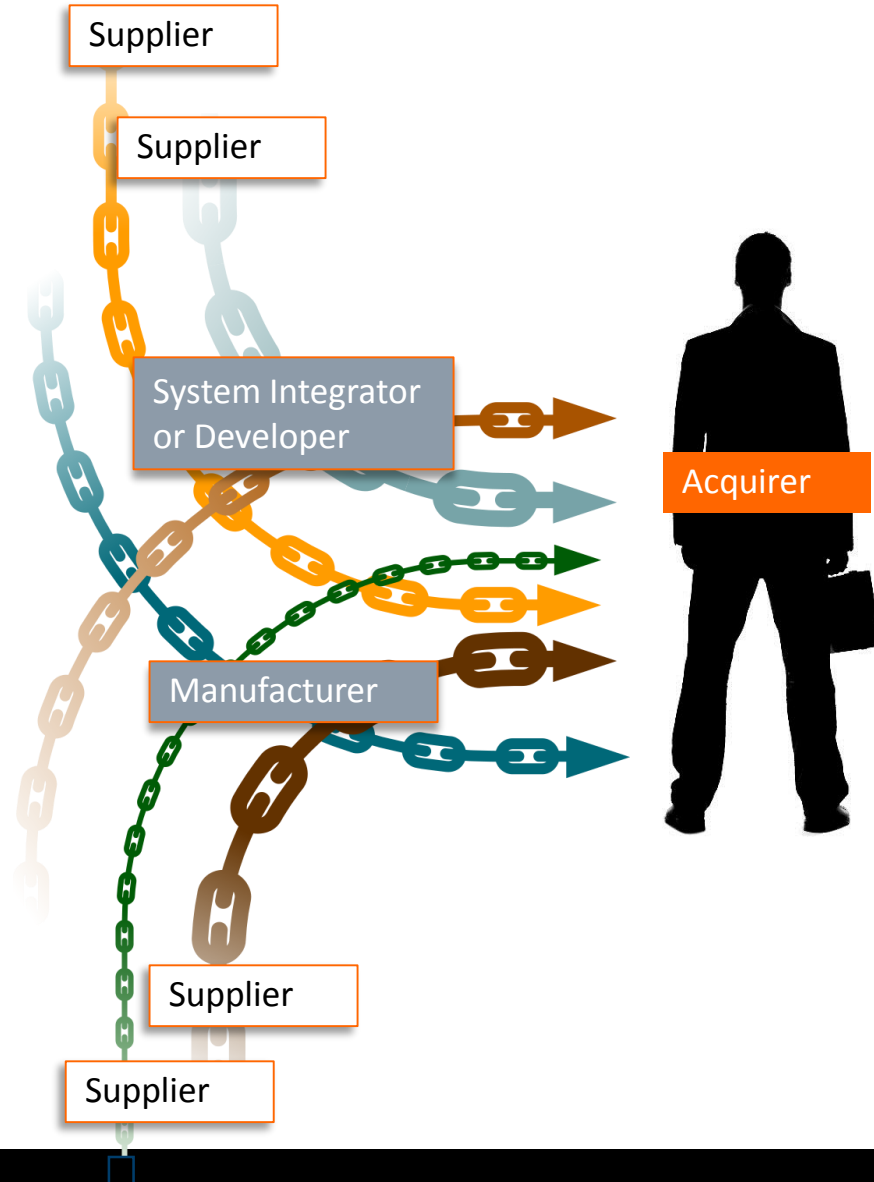
Why

- Organizations can inherit cybersecurity risks from third-party products and services.
- Products inherit cybersecurity risks from third-party components and development frameworks

Benefits

- Provides the basis for improving a organization's supply chain practices
- Identifies potential supply chain risk

Supply Chain Risk Management: *Intentional and Unintentional Acts*



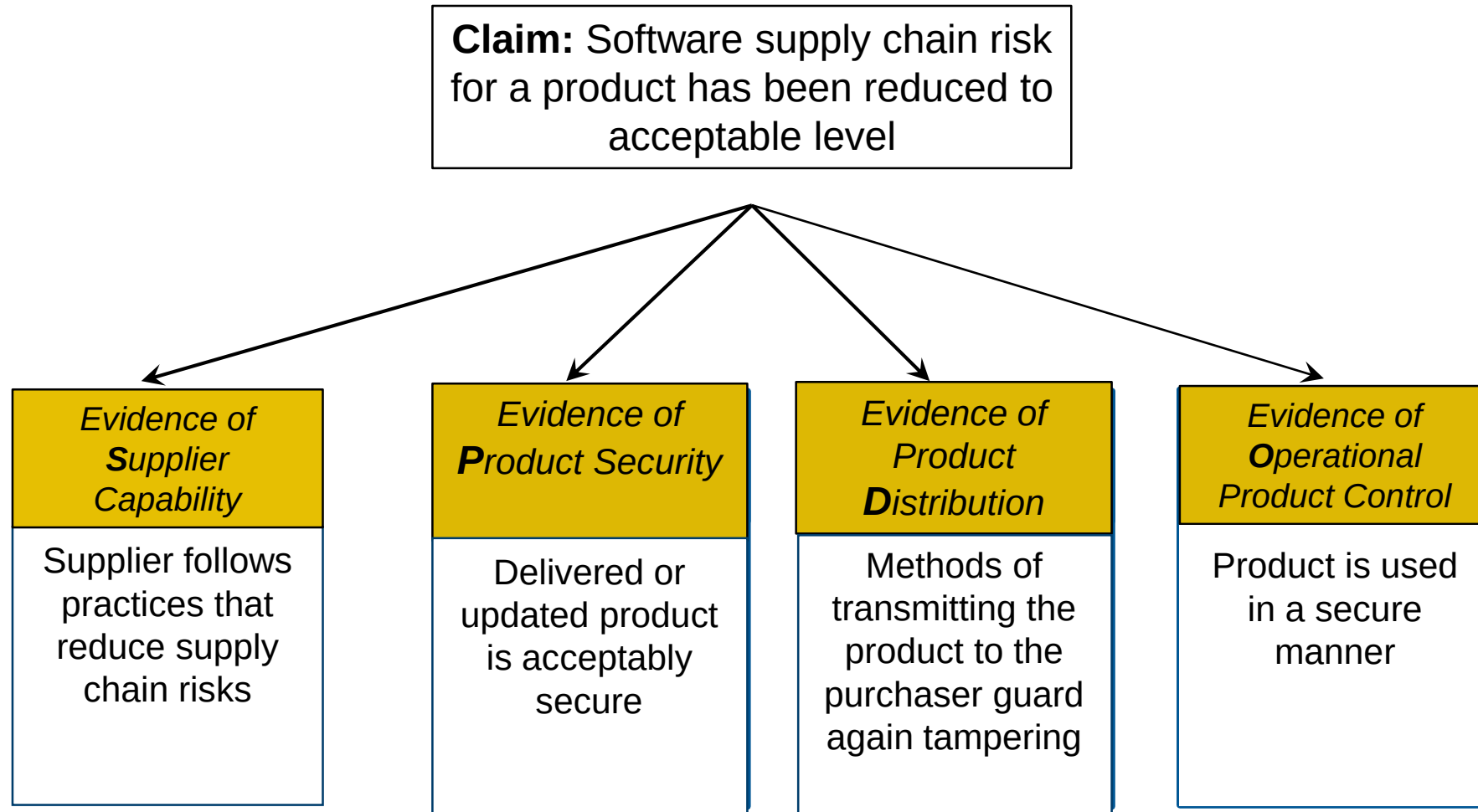
Intentional acts

- counterfeit hardware
- malware insertion

Unintentional acts

- poor code quality
- software vulnerabilities unintentionally inserted

Result: Systems with adverse behaviors



Evaluating and Mitigating Software Supply Chain Security Risks <http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9337>



Evaluate a supplier's security commitment

Evaluate a product's threat resistance

Evaluate product delivery mechanisms

Evaluate the capability to maintain attack resistance during use

SwA for Acquisition

External Dependencies: The Services Supply Chain



Objective: Help critical infrastructure organizations improve their management of external dependencies.

Sponsored by the Department of Homeland Security

Focused on managing the “service supply chain”.

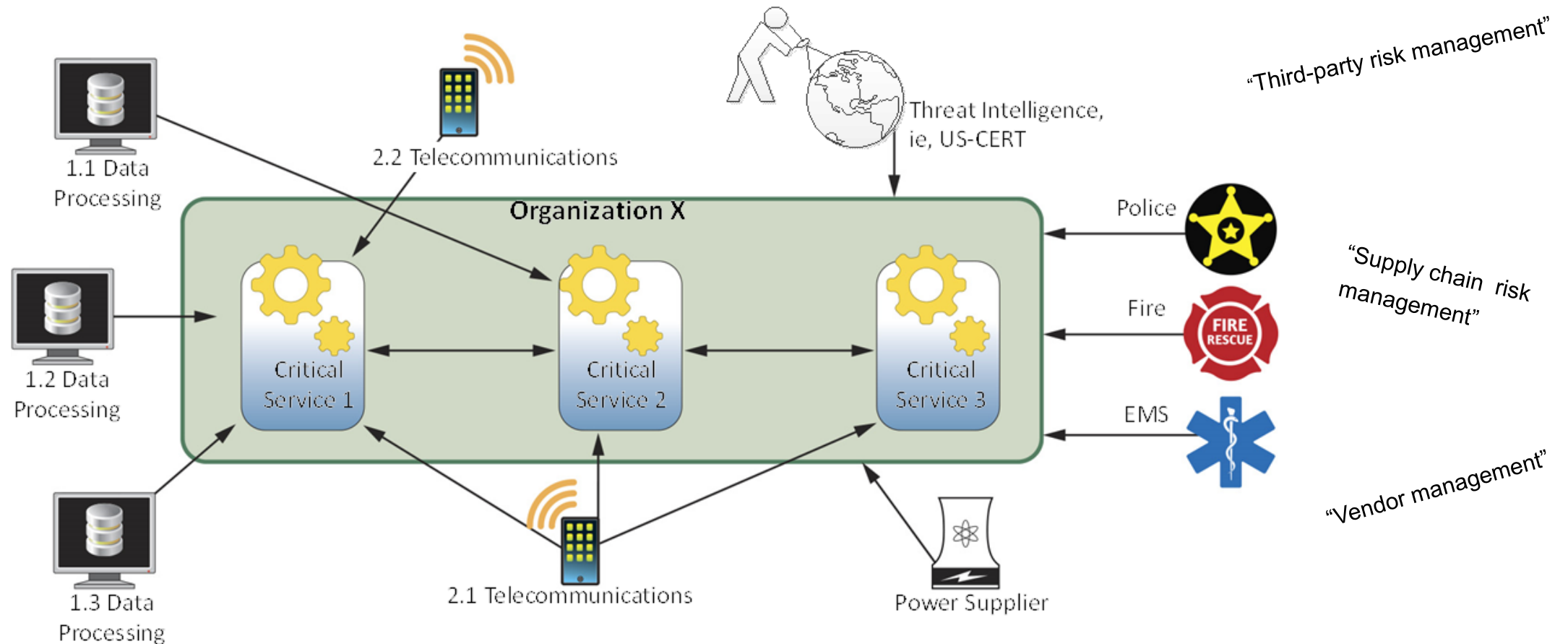
Evaluates the capability of an organization to manage external dependencies and third-party risks. It provides a graphical output to help decision makers select areas for improvement, and links to publicly available resources to provide further guidance.

Additional resources include methods to evaluate and prioritize third-party relationships, and evaluate supplier or third-party cybersecurity and resilience capability.

What do We Mean by External Dependencies Management?

Managing the risk of depending on external entities

External Dependency Management focuses on external entities that provide, support, or operate Information and Communications Technology (ICT).

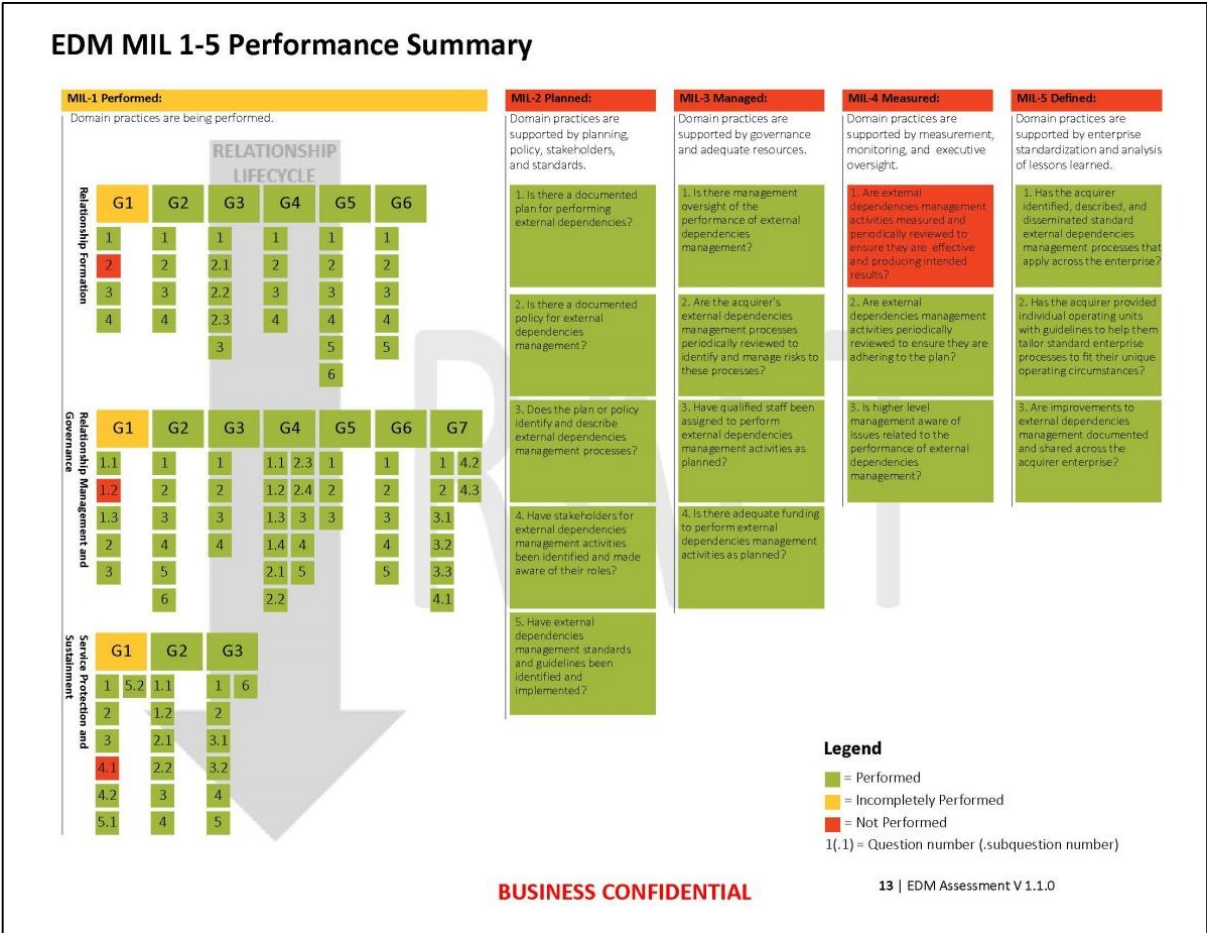
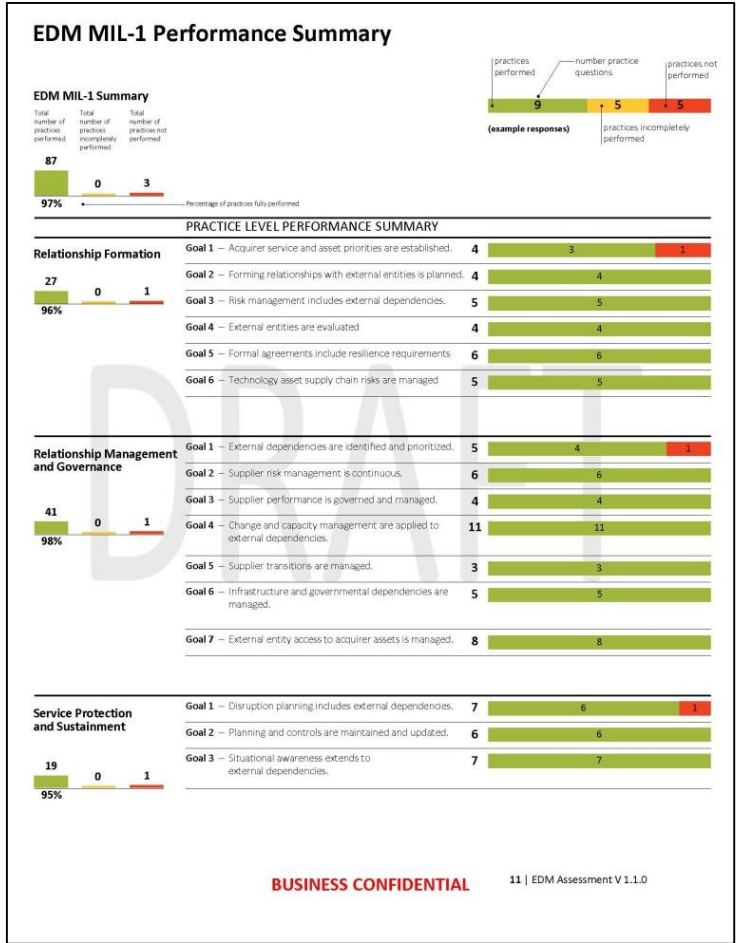


Lifecycle

RF	Relationship Formation The purpose of Relationship Formation is to assess whether the acquirer evaluates and controls the risks of relying on external entities before entering into relationships with them.
RMG	Relationship Management and Governance The purpose of Relationship Management and Governance is to assess whether the acquirer manages ongoing relationships to maintain the resilience of the critical service, and mitigate dependency risk.
SPS	Service Protection and Sustainment The purpose of the Service Protection and Sustainment Domain is to assess whether the acquirer accounts for its dependence on external entities as part of its operational activities around managing incidents, disruptions, and threats.

Graphical Summaries of Results

The Assessment Report provides a graphical overview of an organization’s capabilities and highlights areas that may benefit from further evaluation



SwA for Acquisition

SCRM Standards



ISO/IEC 27036

- multi-part standard offering guidance on the evaluation and treatment of information security risks involved in the acquisition of goods and services from suppliers

NIST 800-53 Rev 4 Assessing Security and Privacy Controls in Federal Information Systems and Organizations: Building Effective Assessment Plans

- state-of-the-practice security controls and control enhancements have been developed and integrated into the catalog addressing such areas as: mobile and cloud computing; applications security; trustworthiness, assurance, and resiliency of information systems; insider threat; **supply chain security**; and the advanced persistent threat.

NIST 800-161 Supply Chain Risk Management Practices for Federal Information Systems and Organizations

SCRM practices build on existing standardized practices in multiple disciplines, which must be practiced at a basic level of maturity

Foundational practices include:

- ensuring that organizations understand the cost and scheduling constraints of implementing SCRM;
- integrating information security requirements into the acquisition process;
- using applicable baseline security controls as one of the sources for security requirements;
- ensuring a robust software quality control process
- establishing multiple sources (e.g., delivery routes for critical system elements).

Reference: NIST 800-161

SwA for Acquisition Summary



Evaluate software supply chain risk in the larger context of

- Supply chain risk
- System risk
- System of systems risk

Risks to a business process or mission thread can come through the software supply chain – include software supply chain risk scenarios in system risk management analysis



Assurance depends on

- the supplier's development capabilities
- security capabilities of the product
- implementation choices by the acquirer
- successfully managing the security risks associated with acquisition requirements

Acquirers can use threat modeling and supply chain risk management to:

- identify potential security risks associated with requirements
- support the creation of supplier selection criteria
- include security risk management expertise to assist in source selections
- understand the security impact of choices

Software Assurance: SwA Metrics





SwA Metrics Challenges

SwA Metrics Research

SwA Metrics

SwA Metrics Challenges



Measuring the software assurance of a product as it functions within a specific system context involves assembling carefully chosen metrics that demonstrate a range of behaviors to establish confidence that the product functions as intended and is free of vulnerabilities

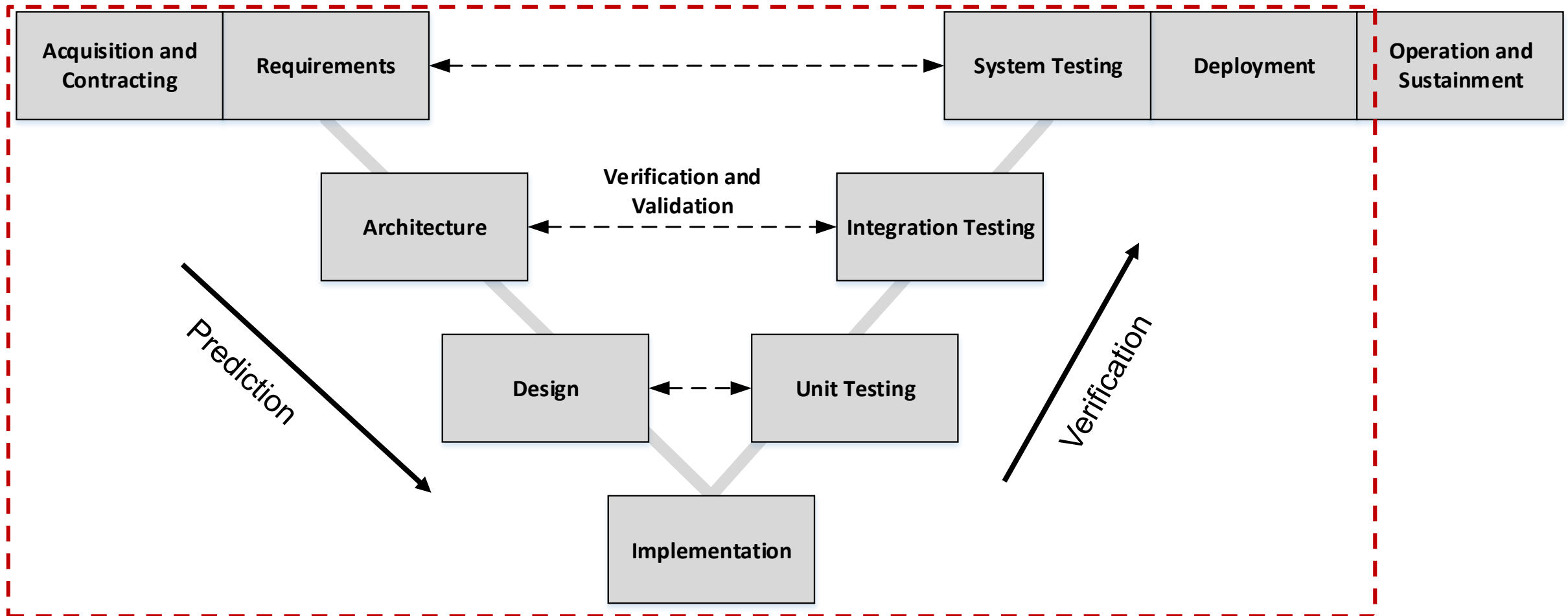
- requirements define the appropriate security behavior
- design addresses these security concerns
- software is never defect free – focus on reducing risk

Metrics should evaluate the expected software assurance as an acquisition or development project is underway so problems can be identified and adjustments can be made to improve the end result

Measures Primarily Focus on Cost & Schedule

Current focus: Will the project finish on time and within budget?

Expanded focus: Will the deployed system have sufficient quality and security?



Properties:

Indicates what is important to success

Measurements should be timely and sufficiently accurate

Actions can be motivated by the indicator

Tenets for Selection:

Tracks a constraint or limit to performance

Predicts future values of the same measure

Predicts future values of another measure

**Examples – process compliance,
requirements volatility, risk exposure**

Key risks to cost, schedule, and quality need to be measured and managed to include security considerations such as:

- requirements volatility, which impacts security design
- defects and vulnerabilities injected and addressed
- architecture weaknesses that become cyclical dependencies, modularity violations, unstable interfaces
- vulnerabilities and weaknesses that become sustainment costs (security technical debt)
- variance from cost estimates, which puts scope or schedule at risk
- failure in requirements or design to anticipate and mitigate various sorts of threat (e.g., known malware)

Establish Transparency of the Security Capability

Questions to be Addressed:

- Will it work in its intended environment?
- What failure rate can be expected?
- Will it be sufficiently secure from attack?

What data do we need to effectively predict operational security?

- Evidence that risk management is applied across the lifecycle
- Evidence that defects are identified, addressed, and reduced
- Evidence that requirements are effectively defined and addressed
- Evidence that defines process compliance, requirements volatility, risk exposure

Today no one organizational group owns software assurance

Integration is needed among software acquisition, cybersecurity, risk management, and SwA.

There will be competing priorities among each SwA participating area, and the mechanisms for open communication and conflict resolution are currently undefined.

Risk management is distributed, but each group uses a disconnected activity. Who is responsible for software risk?

Metrics can be added to each risk process, but the value of multiple stand-alone efforts will be limited.

Who is qualified to address software assurance? Software assurance expertise is needed in all areas integrating with SwA.



Who will be responsible for software assurance (SwA)?

Where in the organization will the SwA capability reside?

Will cybersecurity and SwA capabilities be separated? Why?

How will people acquire and maintain the SwA competencies that they need?

How will the effectiveness of the SwA capability be measured?

SwA Metrics SwA Metrics Research



Establish a shared goal for a focus area

Decompose the goal into sub goals that establish a practice, outcomes and possible metrics (use Goal Question Metric focus - <ftp://ftp.cs.umd.edu/pub/sel/papers/gqm.pdf>)

Pick a starting focus area (e.g. engineering or process improvement or requirements)

Identify a subset of possible metrics that are already available or can be collected with minimal additional effort to begin assembling data for evaluation

Add/adjust metrics building on what you are already doing

Establish a shared goal:

- Example: Applications fielded will have low to medium security risk

Focus on building lifecycle SwA capability integrated into all aspects of acquisition, development and operations to support this goal such as:

- Certification & accreditation includes consideration of SwA and security risk
- Configuration control is effective – knowing that what is certified is what is installed and that these products meet security expectations
- PM, Engineering, and Architecture understand the production targets and plan/design to meet them
- Requirements and architecture incorporate SwA risk mitigations and demonstrate assurance
- Testing – results and processes support application security risk considerations

Engineering Best Practices for Software Security

- **Requirements.** Does the program/project define and manage software security requirements?
- **Architecture.** Does the program/project appropriately address security in its software architecture and design?
- **Implementation.** Does the program/project minimize the number of vulnerabilities inserted into the code?
- **Testing, Validation, and Verification.** Does the program/project test, validate, and verify security in its software components?
- **Support Tools and Documentation.** Does the program/project develop tools and documentation to support secure configuration and operation of software components?
- **Deployment.** Does the program/project consider security during the deployment of software components?

- Attend training for developing security requirements for software (for selected software engineers).
- *Conduct security risk analysis* (includes threat modeling and abuse/misuse cases).
- Define and document software security requirements.
- Conduct reviews (e.g., peer reviews, inspections, and independent reviews) of software security requirements.
- Manage changes to software security requirements.

Activities/Practices	Outputs	Candidate Metrics
Conduct security risk analysis (includes threat modeling and abuse/misuse cases).	Prioritized list of software security risks Prioritized list of design weaknesses Prioritized list of controls/mitigations Mapping of controls/mitigations to design weaknesses	Number and % of software security risks controlled/mitigated (e.g., high and medium risks) Number and % of software security risks accepted/transferred Number and % of software security controls/mitigations selected for requirements development

- Attend training for secure/resilient software architectures (for selected software engineers).
- *Incorporate security requirements into software architecture.*
- Conduct security risk analysis of architecture.
- Address design weaknesses identified during the architectural security risk analysis.
- Conduct security reviews of software architecture (e.g., peer reviews, inspections, and independent reviews).
- Manage security changes to software architecture.

Activities/Practices	Outputs	Candidate Metrics
Incorporate security requirements into software architecture.	Security features in architecture (e.g., authentication, access control, encryption, and auditing) Traceability of software security requirements to security features	Number of applicable security requirements not implemented in software architecture Number of security features without corresponding security requirements Percentage of security requirements addressed by the architecture

- *Secure coding standards are applied.*
- Code developers are trained in the use of secure coding standards.
- Evaluation practices (e.g., code reviews and apply tools) are applied to identify and remove vulnerabilities in delivered code (including code libraries, open source, and other reused components).

Activities/Practices	Outputs	Candidate Metrics
Apply secure coding standards.	Policy that requires the use of secure coding standards Contract language to ensure vendor(s) require use of secure coding standards	Percentage of vendor contracts including requirements for the use of secure coding standards Percentage of system developed using secure coding standards Percentage of code verified for secure coding standard conformance

- *Develop security test cases based on software requirements and risks and issues from prior agency, program, and element experience.*
- Perform a software requirements-based test coverage analysis.
- Perform a software structural test coverage analysis.
- Perform security regression testing on all code impacted by software changes.
- Perform peer security reviews of select test products throughout the software lifecycle.
- Perform independent reviews of select test products throughout the software lifecycle.
- Verify coding standards have been followed.
- Conduct security test readiness reviews as part of a Test Readiness Review (TRR).
- Perform operational security testing for the integrated system.

Activities/Practices	Outputs	Candidate Metrics
Develop security test cases based on software requirements and risks and issues from prior agency, program, and element experience.	Security-related test cases based on software requirements, risks, and prior lessons learned Policy level and legal requirements included in test cases Requirements Traceability and Verification Matrix (RTVM)	Security software requirements in test spec Security requirements tested • percent passed without issues • percent passed with issues • percent failed • percent tests to be rerun Number of test cases Average number of test cases per program/function (normalized by size, function, function point, or other) Defect rates • total number of defects • percentage by criticality (low, medium, high) • average time to correct

Activity progress, outputs, and metrics should be reviewed and evaluated at each engineering review.

- **Initial Technical Review (ITR).** Assess the capability needs (including security) and materiel solution approach.
- **Alternative Systems Review (ASR).** Ensure that solutions will be cost effective, affordable, operationally effective, and can be developed in timely manner at an acceptable level of software security risk.
- **System Requirements Review (SRR).** Ensure that all system requirements (including security) are defined and testable, and consistent with cost, schedule, risk (including software security risk), technology readiness, and other system constraints.
- **Preliminary Design Review (PDR).** Evaluate progress and technical adequacy of the selected design approach.
- **Critical Design Review (CDR).** Determine that detail designs satisfy the design requirements (including software security) established in the specification and establish the interface relationships.

Software Assurance Course Summary



Development

- Cybersecurity Lifecycle
- Requirements Engineering (e.g., SQUARE)
- Threat Modeling (e.g., STRIDE)
- Architectural Analysis
- Code Analysis
- Quality Improvement integrated with security

Mission Assurance

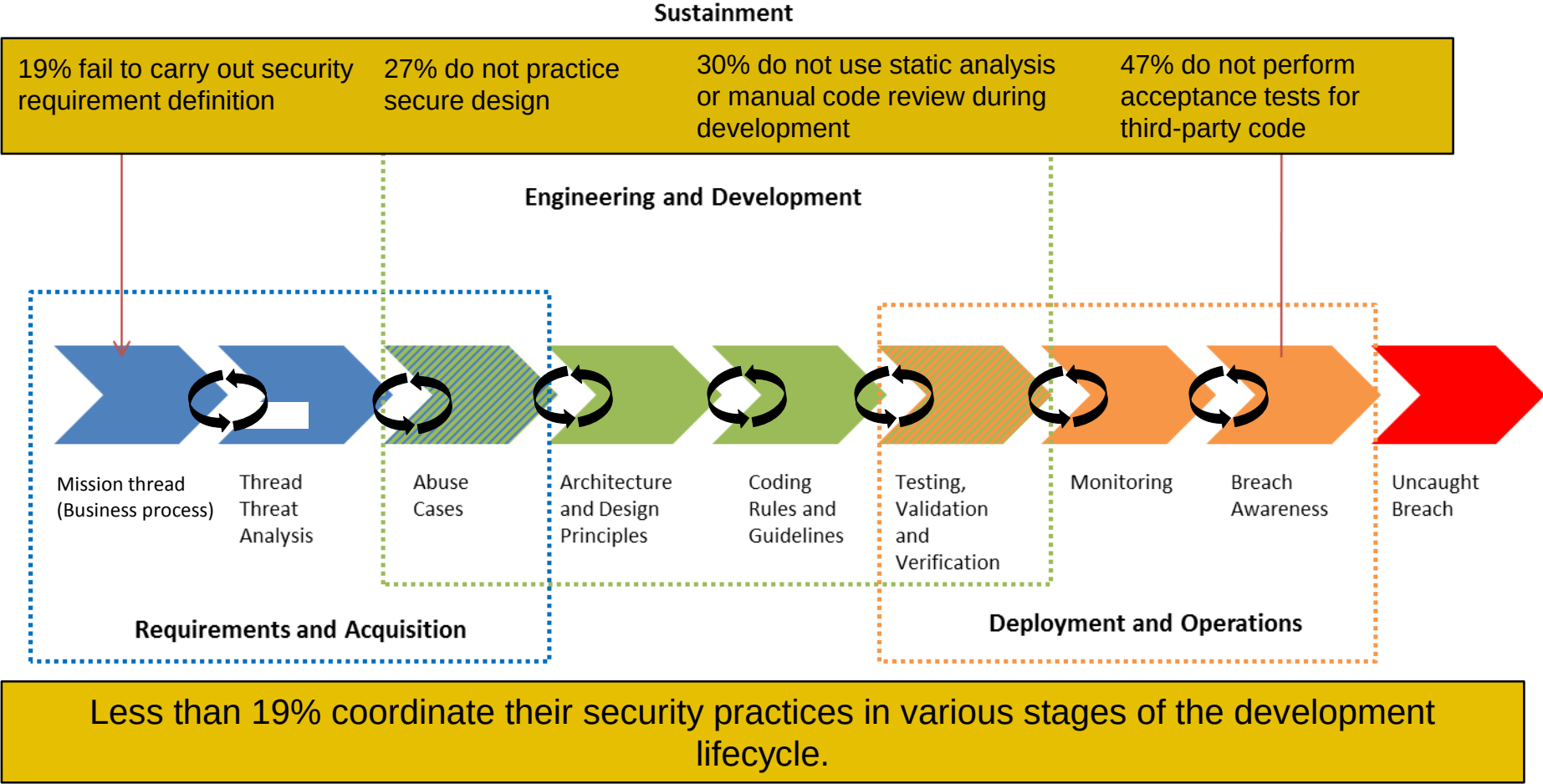
- Security Engineering Risk Analysis (SERA)

Acquisition

- Software Supply Chain Risk Management
- Supplier Relationship Management
- SCRM Standards

Measurement (add security considerations to cost and schedule)

Opportunities for Improvement



Source: Forrester Consulting, "State of Application Security," January 2011

Course Summary

Additional References



Books

- *The Security Development Lifecycle: SDL: A Process for Developing Demonstrably More Secure Software* (Howard, Lipner) “Threat Modeling” chapter
- *Threat Modeling: Designing for Security* (Adam Shostack)

MSDN Magazine

- Reinvigorate your Threat Modeling Process
- Threat Modeling: Uncover Security Design Flaws Using The STRIDE Approach

Article

- Experiences Threat Modeling at Microsoft

Establishing the Importance of System Context for Software Assurance



"The Impact of Contextual Factors on the Security of Code" in CrossTalk's May/June 2015 issue titled "What is Software Engineering?"

"Model-Based Engineering for Supply Risk Chain Management" in CrossTalk's Sept/Oct 2015 issue titled "Supply Chain Assurance"

Alberts, C.; Woody, C.; & Dorofee, A. *Evaluating Security Risks using Mission Threads* (CMU/SEI-2014-TN-025). Software Engineering Institute, Carnegie Mellon University, 2014.
http://resources.sei.cmu.edu/asset_files/TechnicalNote/2014_004_001_427329.pdf

Woody, C.; & Alberts, C. "Evaluating Security Risks using Mission Threads." *CrossTalk* 10, 2 (September/October 2014): 14-19.
<http://www.crosstalkonline.org/storage/issue-archives/2014/201409/201409-Woody.pdf>

Alberts, C.; Woody, C.; & Dorofee, Wireless Emergency Alerts Commercial Mobile Service Provider (CMSP) Security Guidelines,
<http://www.firstresponder.gov/TechnologyDocuments/Wireless%20Emergency%20Alerts%20CMSP%20Cybersecurity%20Guidelines.pdf>

Supply Chain Further Reading

Axelrod, C. Warren, "Mitigating Software Supply Chain Risk," ISCA Journal Online, Vol 4., 2013, <https://www.isaca.org/journal/archives/2013/volume-4/pages/jonline-mitigating-software-supply-chain-risk.aspx>

Axelrod, C. Warren, "Malware, Weakware and the Security of Software Supply Chains," Cross-Talk, March/April 2014, p. 20, <http://static1.1.sqspcdn.com/static/f/702523/24448961/1393725140627/201403-Axelrod.pdf?token=XW1qPmSalR9Br6DRtLAqn76DwXk%3D>

Ellison, Robert et al, *Software Supply Chain Risk Management: From Products to Systems of Systems*, Software Engineering Institute, Dec 2010, <http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=9377>

Ellison, Robert et al. *Evaluating and Mitigating Software Supply Chain Security Risks*, Software Engineering Institute, May 2010, http://resources.sei.cmu.edu/asset_files/technicalnote/2010_004_001_15176.pdf

Ellison, Robert and Woody, Carol, "Supply-Chain Risk Management: Incorporating Security into Software Development," Proceedings of the 43rd Hawaii International Conference on System Sciences, 2010, <http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=297335>

Jarzombek, Joe, "Collaboratively Advancing Strategies to Mitigate Software Supply Chain Risks," July 30, 2009, http://csrc.nist.gov/groups/SMA/ispab/documents/minutes/2009-07/ispab_july09-jarzombek_swa-supply-chain.pdf

"Software Supply Chain Risk Management & Due-Diligence," Software Assurance Pocket Guide Series: Acquisition & Outsourcing, Vol II, Version 1.2, June 16, 2009, <http://osgug.ucaiug.org/utilisec/Shared%20Documents/Security%20Procurement%20Guidelines%20and%20Language/DHS%20-%20Building%20Security%20In%20-%20Software%20Supply%20Chain%20Risk%20Management%20and%20Due-Diligence.pdf>

Third Party Software Security Working Group, "Appropriate Software Security Control Types for Third Party Service and Product Providers," Financial Services Information Sharing and Analysis Center, 2013, http://docs.ismgcorp.com/files/external/WP_FSISAC_Third_Party_Software_Security_Working_Group.pdf

Young, Lisa, SEI Cyber Minute: CERT Resilience Management Model (RMM), 2016, <http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=485774>

State of the Practice

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

Quantifying Uncertainty in Early Lifecycle Cost Estimation (QUELCE)

http://insights.sei.cmu.edu/sei_blog/2012/06/quantifying-uncertainty-in-early-lifecycle-cost-estimation-quelce-an-update.html

BBN research

YouTube: https://youtu.be/kc8-_rJAo94