Software Solutions Symposium 2017

March 20–23, 2017

Security Measurement: Establishing Confidence that Security Is Sufficient

Carol Woody, Ph.D.

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

[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.

Copyright 2017 Carnegie Mellon University

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense.

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 may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

DM-0004404

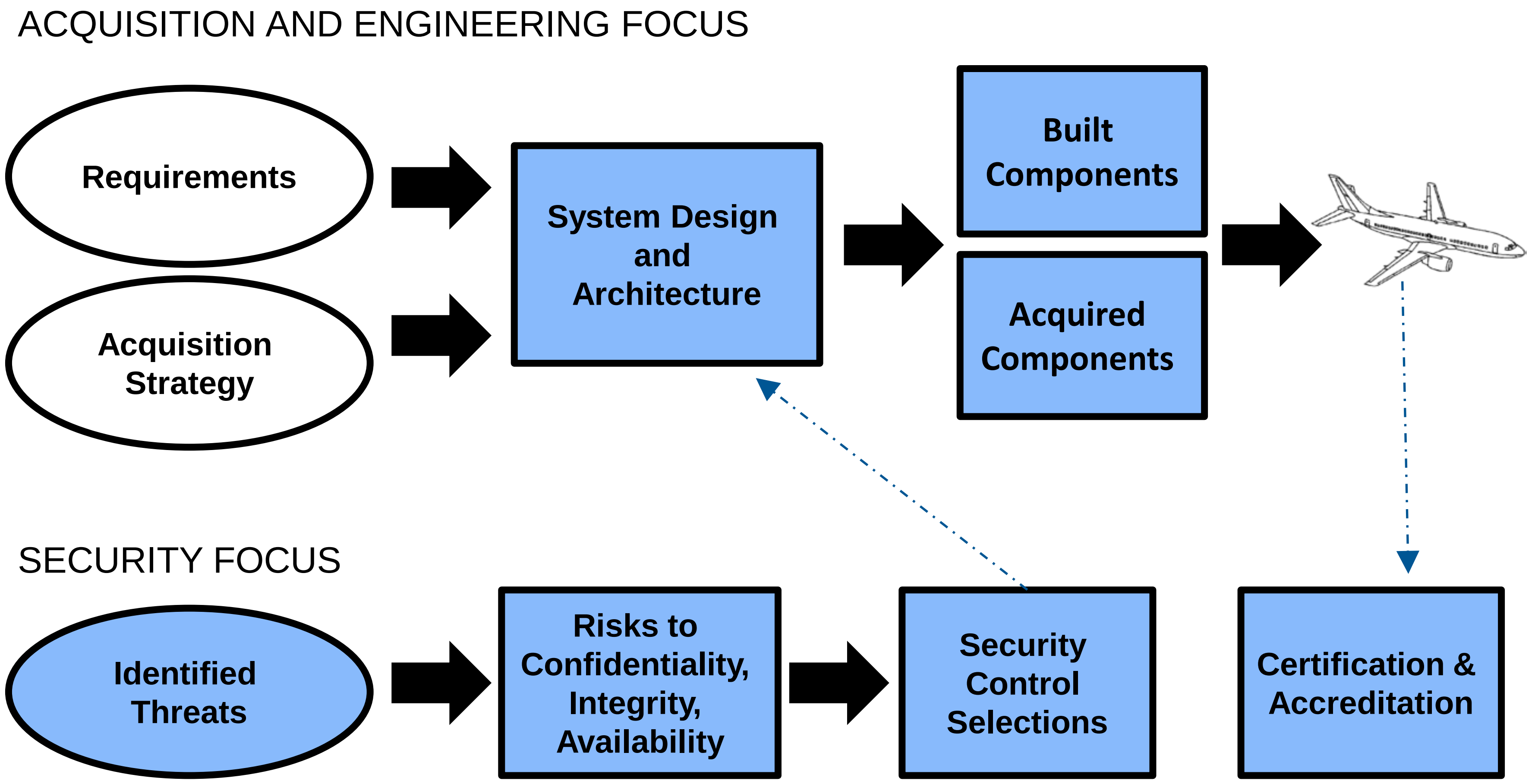
Software Engineering Institute | Carnegie Mellon University

Security Measurement: Establishing Confidence that Security Is Sufficient
March 20–23, 2017
© 2017 Carnegie Mellon University

[[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.

2

Is the System Secure?


ACQUISITION AND ENGINEERING FOCUS
Requirements
Acquisition Strategy
System Design and Architecture
Built Components
Acquired Components
SECURITY FOCUS
Identified Threats
Risks to Confidentiality, Integrity, Availability
Security Control Selections
Certification & Accreditation

[[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.

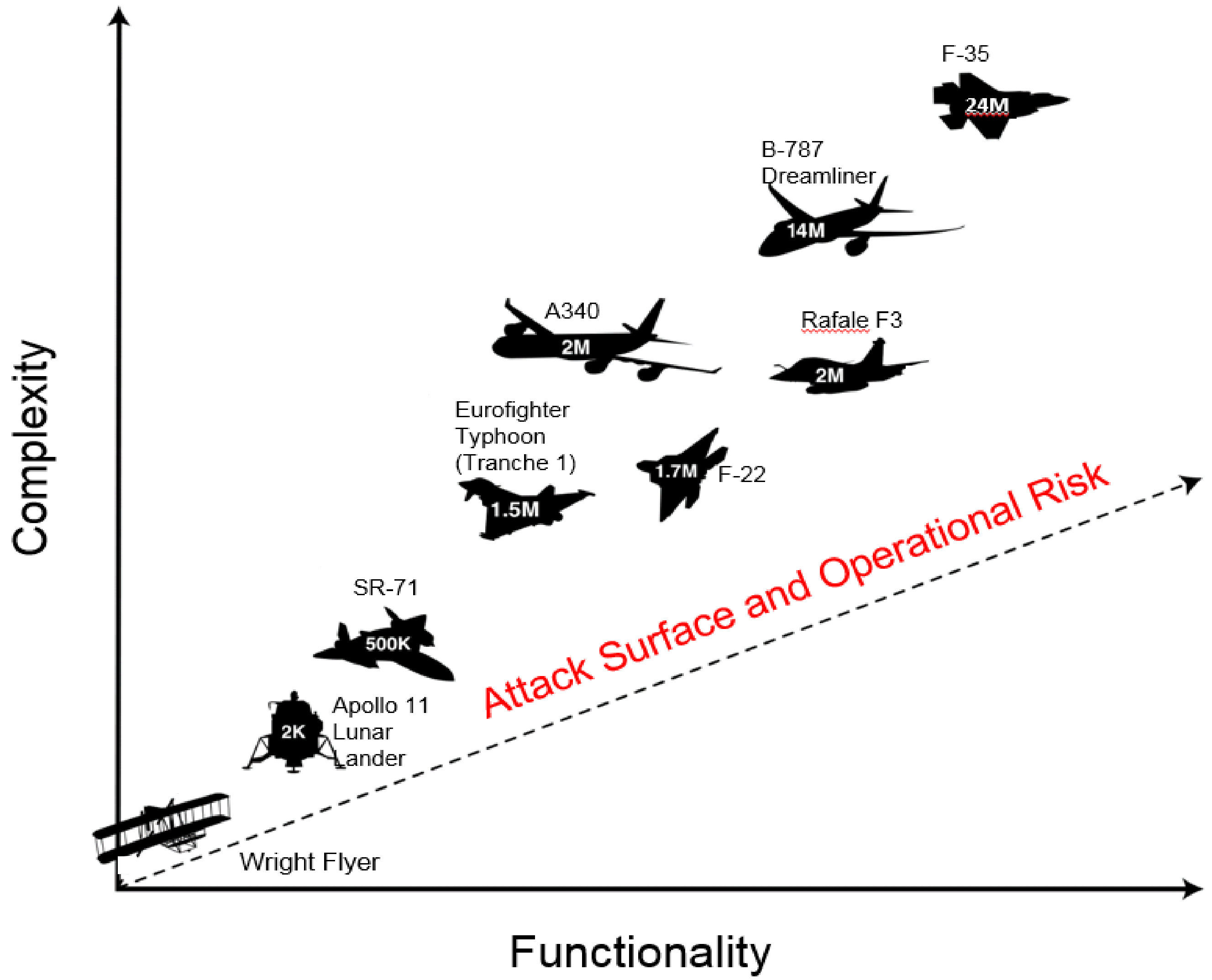
The DoD (Along with Everyone Else) Is in the Software Business

The report *Critical Code: Software Producibility for Defense*, issued in 2010, states that "software has become essential to all aspects of military system capabilities and operations."

- 1960: Software handled 8% of the F-4 Phantom fighter's functionality.
- 1982: Software handled 45% of the F-16 Fighting Falcon's functionality.
- 2000: Software handled 80% of the F-22 Raptor's functionality.

[[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.

Increased Software and Complexity -> More Security Issues



[[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.

Estimating Software Vulnerabilities

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

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)

[[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.

Software Solutions
Symposium 2017

Security Measurement: Establishing Confidence
that Security Is Sufficient

Using Engineering Evidence to Reduce Fear, Uncertainty, and Doubt (FUD) for Software Security

Software Can Be Measured Extensively

Many metrics are collected about software in a wide range of areas:

- security requirements metrics
  - security controls for confidentiality, integrity, and availability
  - use and abuse cases
- threat, vulnerability, and risk metrics
- quality and complexity metrics
- security compliance metrics
- resiliency metrics
- process metrics
- technical debt metrics
- verification and validation metrics

[[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.

Engineering Measurement Questions

Are we measuring the right things at the right time?

Are we moving in the right direction?

Do we collect information soon enough to react to problems?

[[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.

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

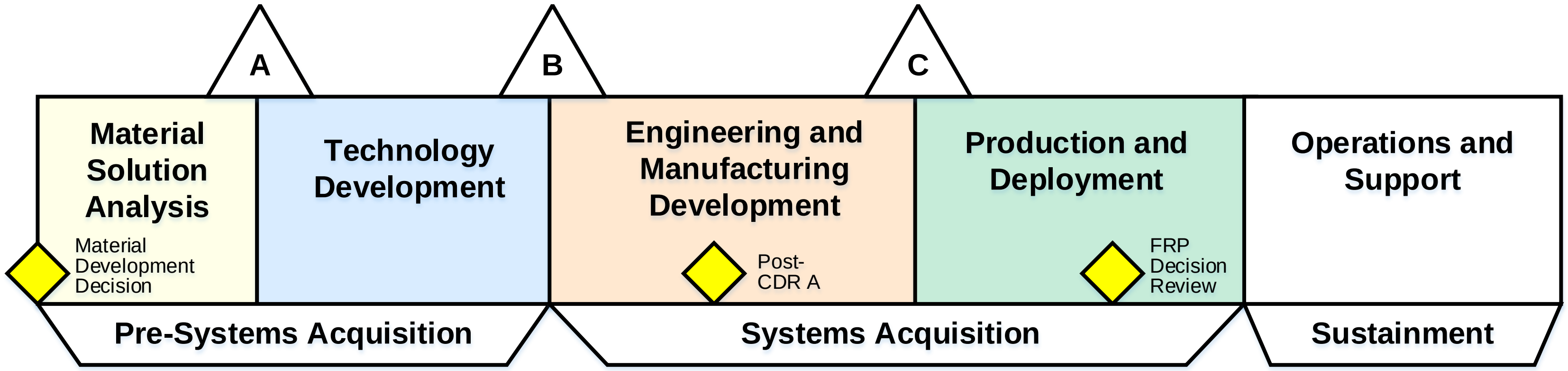
Single-seat, single-engine, all-weather stealth multirole

- 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

[[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.

Engineering Is Applied Across the Lifecycle

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.

[[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.

Software Solutions Symposium 2017

Security Measurement: Establishing Confidence that Security Is Sufficient

Engineering for Software Security

Engineering the System Software to Be Secure

Establish engineering requirements for software security.

Follow best engineering practices for software security.

Collect evidence that engineering practices are being addressed.

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

© 2017 Carnegie Mellon University

[[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.

Establish Engineering Requirements for Software Security

Example for the Airplane

Mission- and flight-critical applications executing on the plane or used to interact with the plane from ground stations have low cybersecurity risk exposure.

Risk exposure can be based on estimated probability and impact

Risk Exposure Matrix

		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)

[[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.

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?

[[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.

Software Security Requirements Practices

- 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.

[[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.

Software Security Requirements Metrics

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

© 2017 Carnegie Mellon University

[[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.

Software Security Architecture Practices

- 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.

[[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.

Software Security Architecture Metrics

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

[[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.

Software Security Implementation Practices

- *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).

[[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.

Software Security Implementation Metrics

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

© 2017 Carnegie Mellon University

[[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.

Software Security Testing, Validation, and Verification Practices - 1

- *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.

[[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.

Software Security Testing, Validation, and Verification Practices - 2

- 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.

[[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.

Software Security Testing, Validation, and Verification Metrics

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

[[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.

Engineering Reviews Confirm Security Progress

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.

© 2017 Carnegie Mellon University

[[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.

Software Solutions Symposium 2017

Security Measurement: Establishing Confidence that Security Is Sufficient

Building Blocks for Engineering Software Security

Software Assurance Framework (SAF)

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

[[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.

Key Practice Areas for Software Security

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

[[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.

Goal Question Metric Approach

Developed in the 1980s as a structuring mechanism to establish a line-of-sight from a goal to appropriate metrics

- Well-recognized and widely used metrics approach
- Useful results, but by no means comprehensive
- See *The Goal Question Metric Approach* (ftp://ftp.cs.umd.edu/pub/sel/papers/gqm.pdf)

[[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.

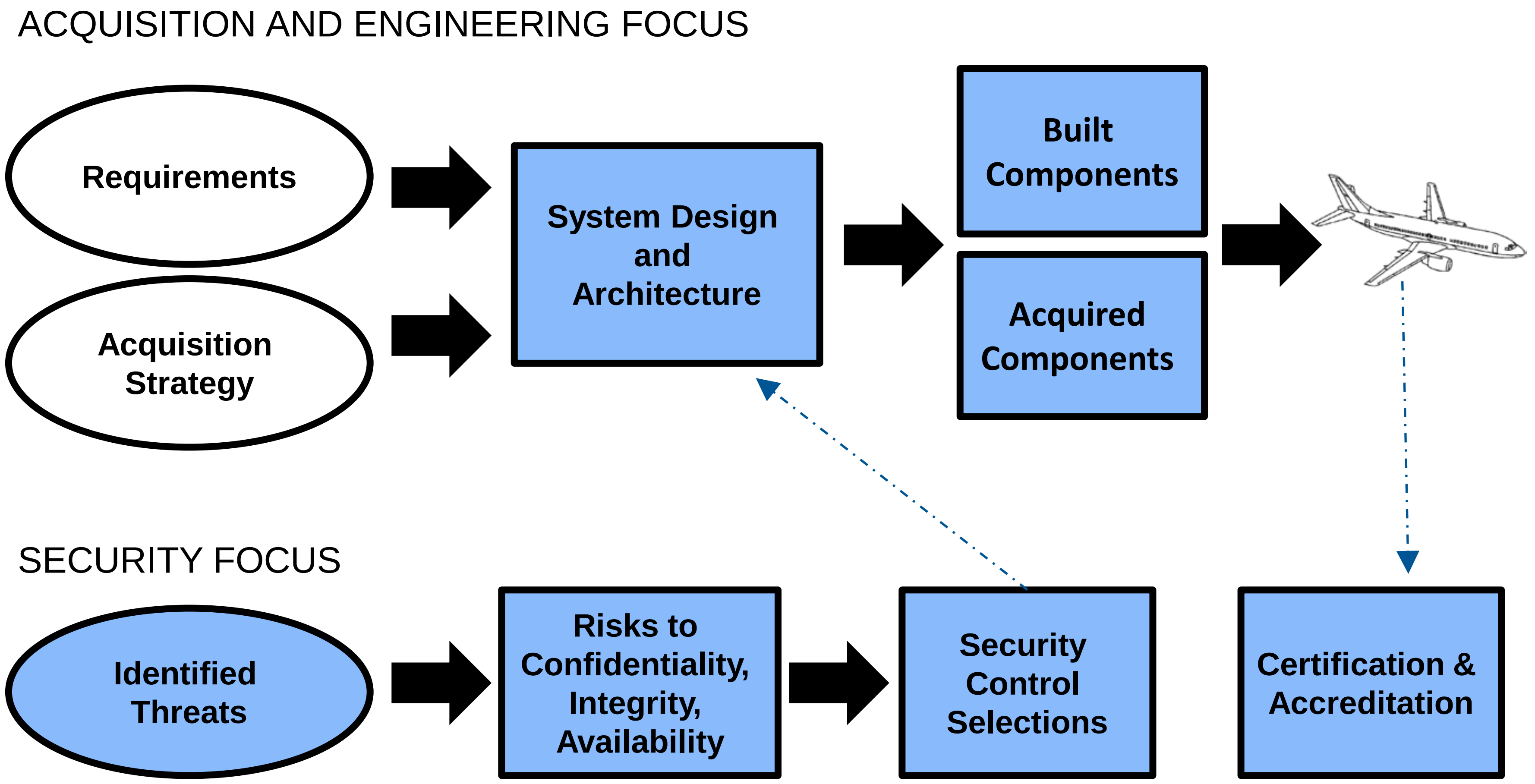
Software Solutions Symposium 2017

Security Measurement: Establishing Confidence that Security Is Sufficient

Summary

Is the System Secure?

Instead of just collecting information about the final product…


ACQUISITION AND ENGINEERING FOCUS
Requirements
Acquisition Strategy
System Design and Architecture
Built Components
Acquired Components
SECURITY FOCUS
Identified Threats
Risks to Confidentiality, Integrity, Availability
Security Control Selections
Certification & Accreditation

© 2017 Carnegie Mellon University

[[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.

Implement Software Security Engineering

Collect and evaluate engineering evidence about product, processes, and practices to show the system appropriately addresses software security across the lifecycle.

- Define the software security requirement(s) for the system.
- Select the metrics that answer the questions that engineering needs to provide evidence that the requirements are addressed:
  - requirements
  - architecture
  - implementation
  - testing, validation, and verification
  - support tools and documentation
  - deployment
- Incorporate metrics reporting and evaluation in all engineering technical reviews.

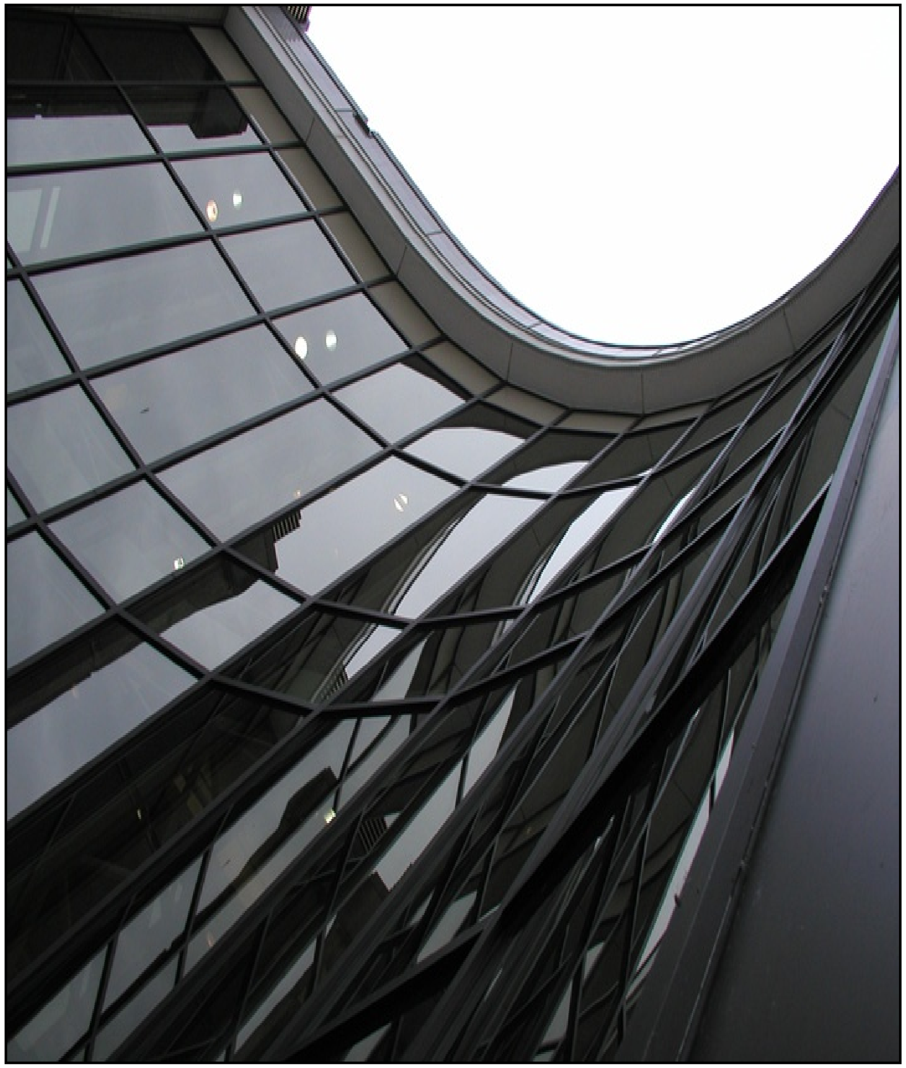
© 2017 Carnegie Mellon University

[[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.

Contact Information

Carol Woody
cwoody@cert.org

Web Resources (CERT/SEI)
http://www.sei.cmu.edu/



[[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.