Engineering Emergence in Systems of Systems: Software and the Growth of Insecurity

Carol Woody, Ph.D.
Technical Manager,
Cybersecurity Engineering

Software Engineering Institute (SEI)
Carnegie Mellon University (CMU)
Pittsburgh, PA 15213

Notices

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.

DM17-0628

Software in Systems of Systems - 1

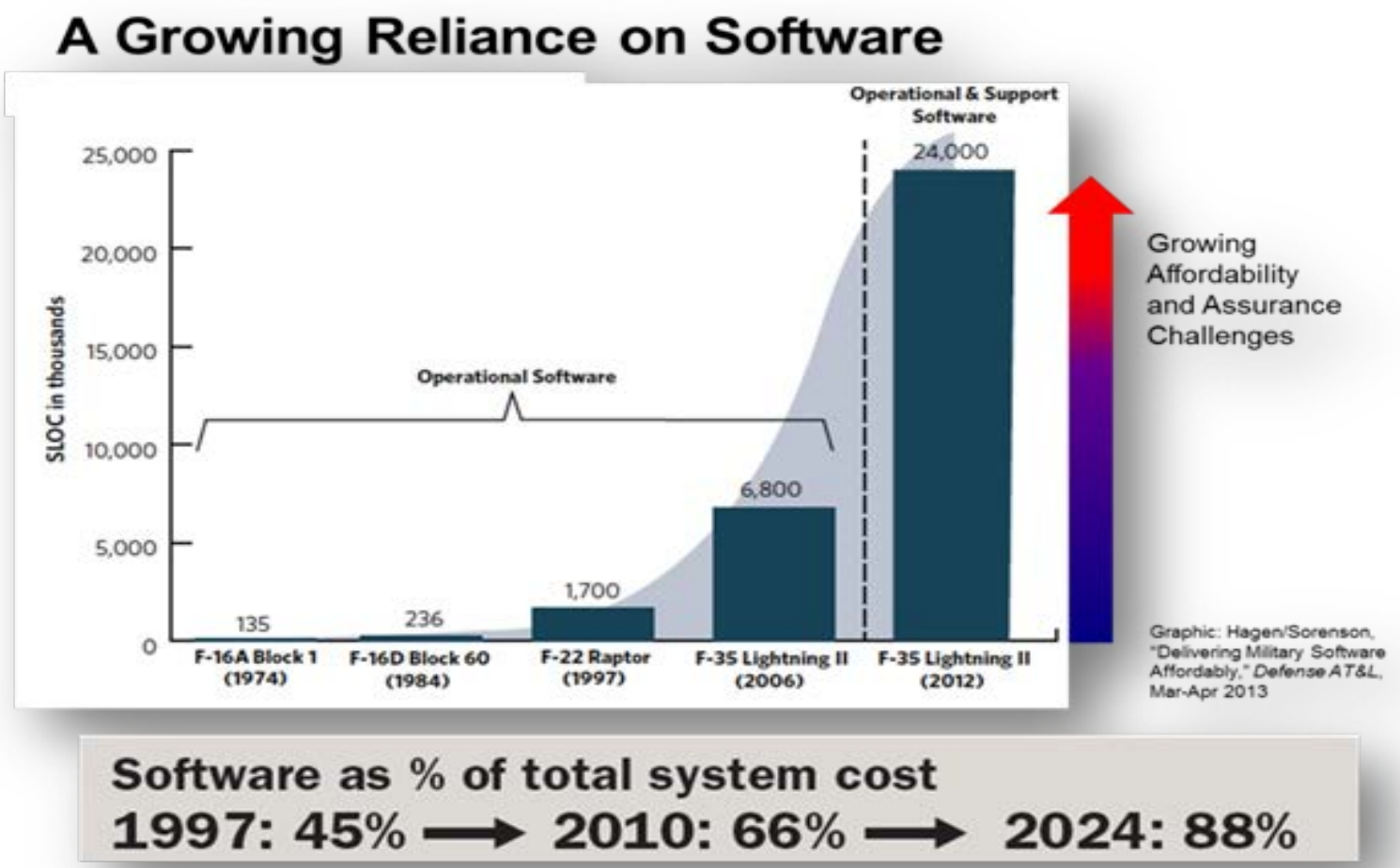
SoS Characteristic (Maier 1998)	Growing Insecurity	Engineering Software to be Secure
Operational Independence	Acquirers/Integrators assemble software from many vendors to seamlessly deliver end-to-end mission capability	Acquirers must identify and mitigate vulnerabilities in software performing mission-critical functions
Managerial Independence	Vendors build and sell software for specialized niche markets (e.g. point-of-sales, printing, Cloud computing)	Acquirers select market dominants (costs more widely distributed, more resources for support, more functionality growth)
Evolutionary Development	Vendors release new functionality to capture market share and drop support of older versions	Acquirers must patch critical software quickly to reduce the attack potential

Software in Systems of Systems - 2

SoS Characteristic (Maier 1998)	Growing Insecurity	Engineering Software to be Secure
Emergent Behavior	Vendors drive down costs through standardized interfaces (e.g. TCP/IP), reuse and push for early releases to dominate their niche markets; Vendor demand licenses that absolve them of liabilities Acquirer's focus on least cost and speed of delivery with extensive connectivity results in widespread vulnerability	Acquirers must impose and monitor quality and security related requirements in their vendor contracts and ensure vendors manage their software supply chains effectively (increased costs and increased oversight)
Geographic Distribution	Vendors deliver insecure-by-default software (faster and easier)	Acquirer must impose secure-by-default requirements

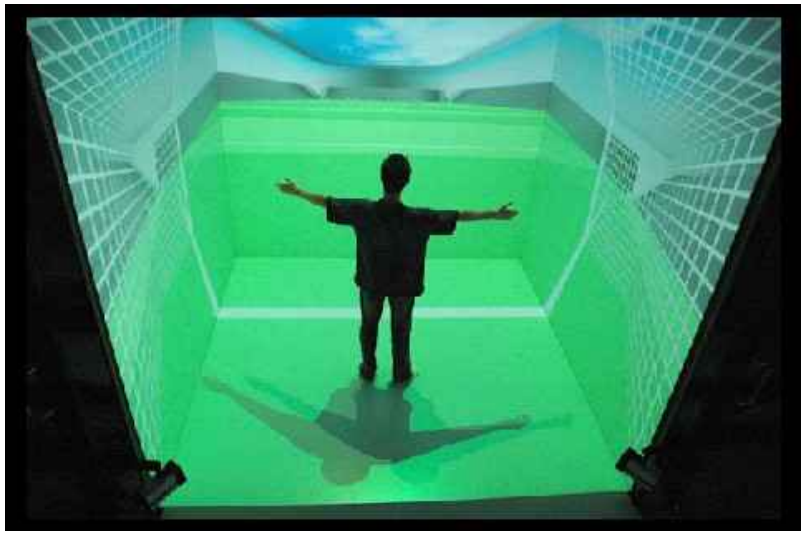
Critical Software-driven Changes in the Technology Landscape

Software Reliance is Rapidly Expanding



Source: U.S. Air Force Scientific Advisory Board. *Sustaining Air Force Aging Aircraft into the 21st Century* (SAB-TR-11-01). U.S. Air Force, 2011.

Software is the New Hardware



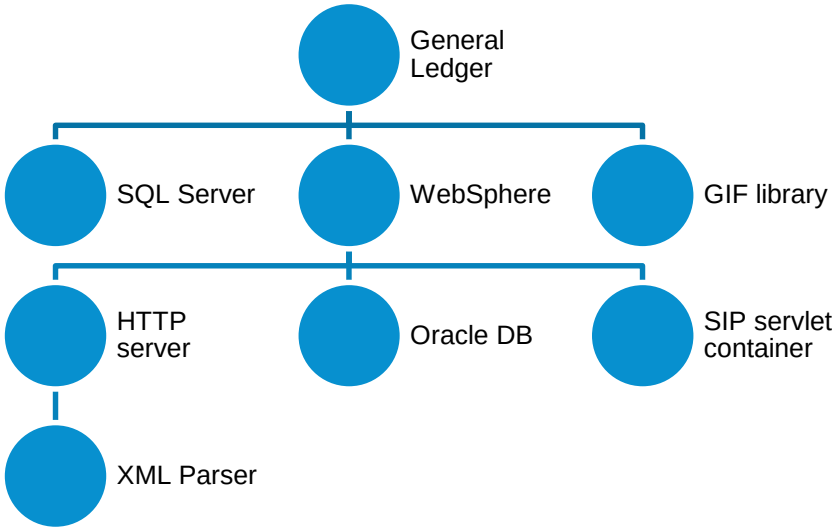
Information Technology (IT) is moving from specialized hardware to software, virtualized as

- Servers: virtual Central Processor Units (CPUs)
- Storage: Storage Area Networks (SANs)
- Switches: Soft switches
- Networks: Software defined networks

Scalable cloud computing environments are replacing organization-owned data centers

Firmware, which can be updated, provides the low-level program control for hardware

Development is now Assembly

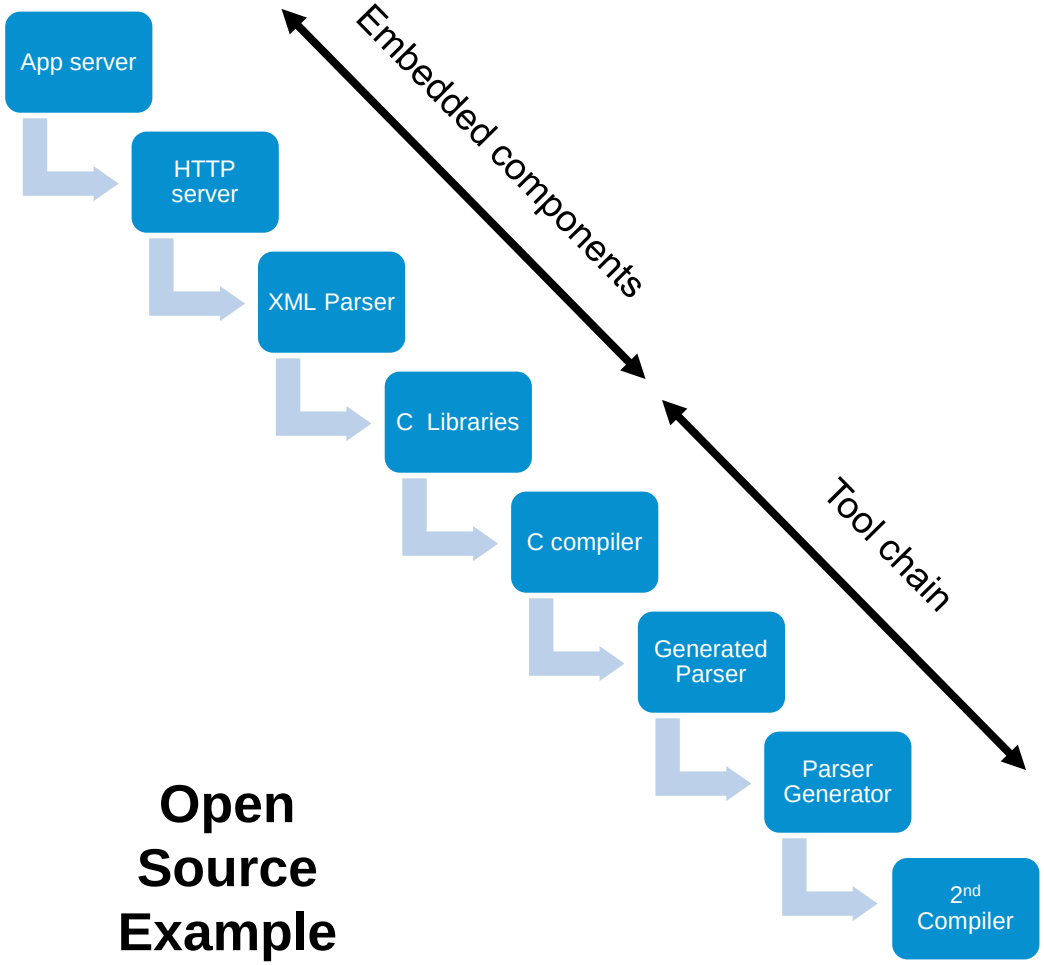


Note: hypothetical application composition

Collective development – context:

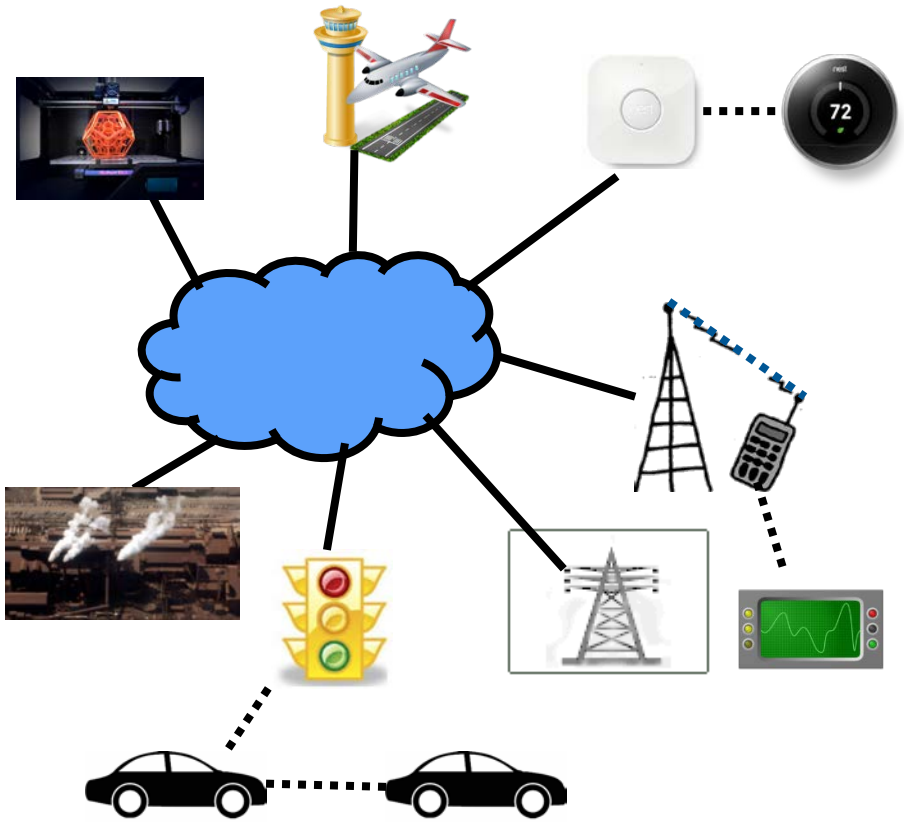
- Too large for single organization to support
- Too much specialization
- Too little value in individual components
- Growing shift to open source
- Each component collects, stores, and sends data in different file structures and formats

Software Sources are Many, Varied, Reusable



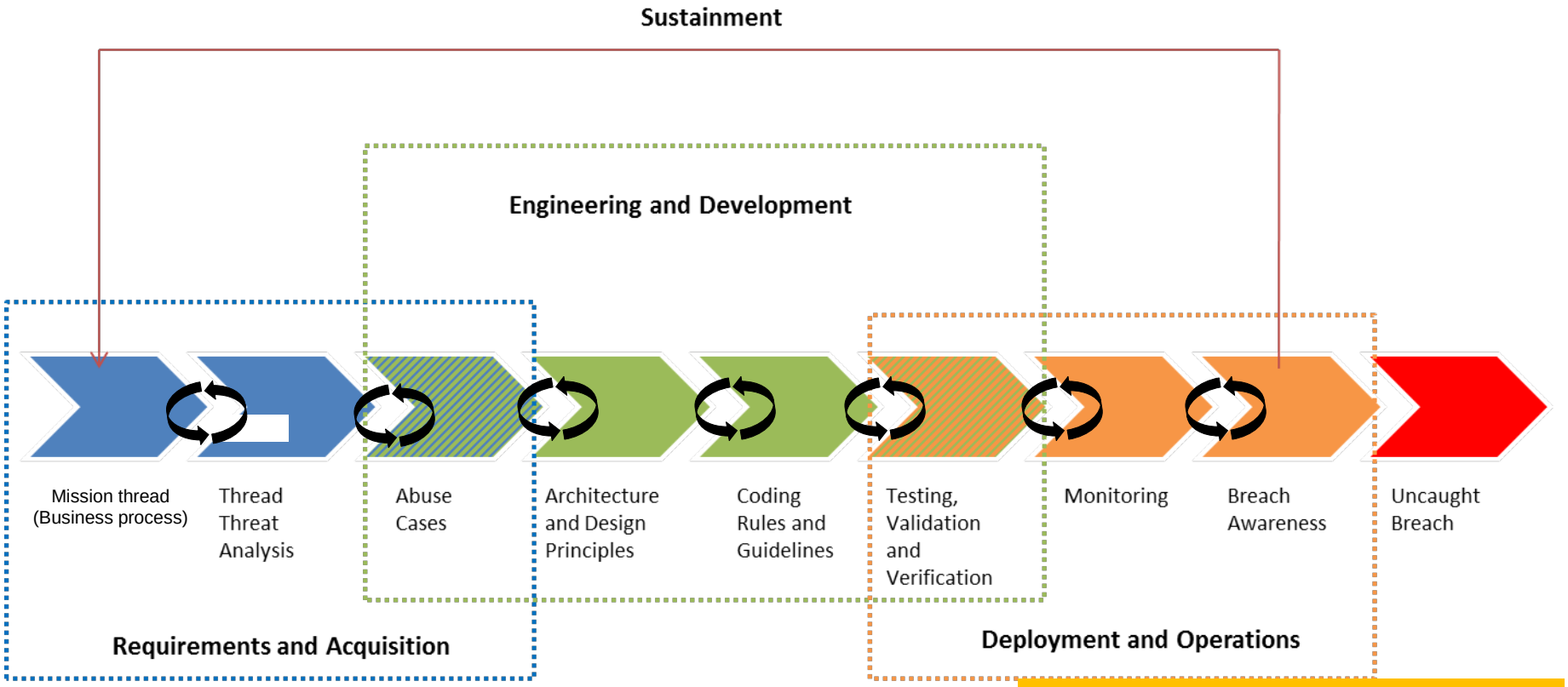
Open Source Example

Software Connecting and Communicating Grows



- 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

Security Is a Lifecycle Challenge



Design Weaknesses

Coding Weaknesses

705 Common Weaknesses

CWE.Mitre.org

89,888 Common Vulnerability Enumerations (CVE)

CVE.Mitre.org

Security Vulnerabilities are Increasing

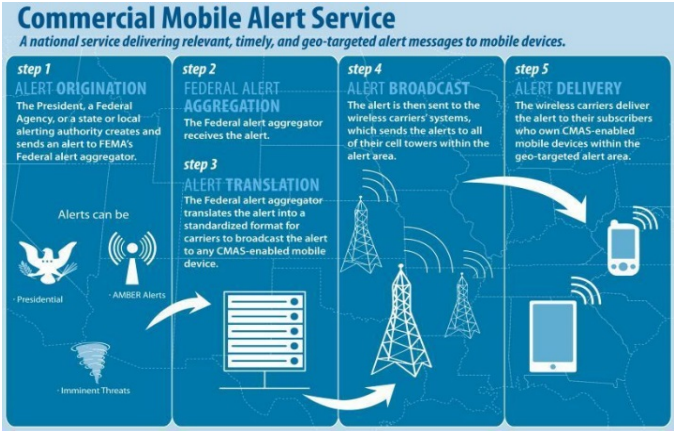
Definition: Security vulnerability is a weakness which allows an attacker to bypass security controls

Requires three elements:

- System susceptibility or flaw,
  - Millions of lines of software code handling an ever increasing amount of functionality
  - Thousands of software vulnerabilities
  - Increased reliance on commercial and open source software
- Attacker access to the flaw, and
  - Increased connectivity linking systems to other systems and connecting to new types of devices (Internet of Things)
  - Increased system and device remote communication capability
- Attacker capability to exploit the flaw
  - Access to the same tools and techniques used to build software
  - Reverse engineering capabilities for commercial and open source
  - Malware and attack platforms and frameworks

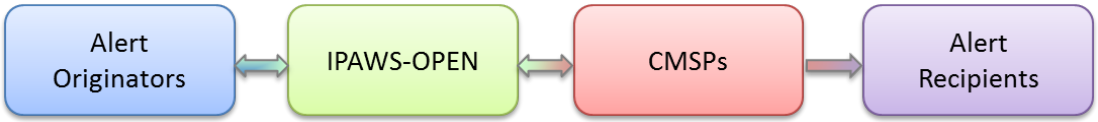
SEI Interest in Emergence

SEI WEA Research 2011-2016



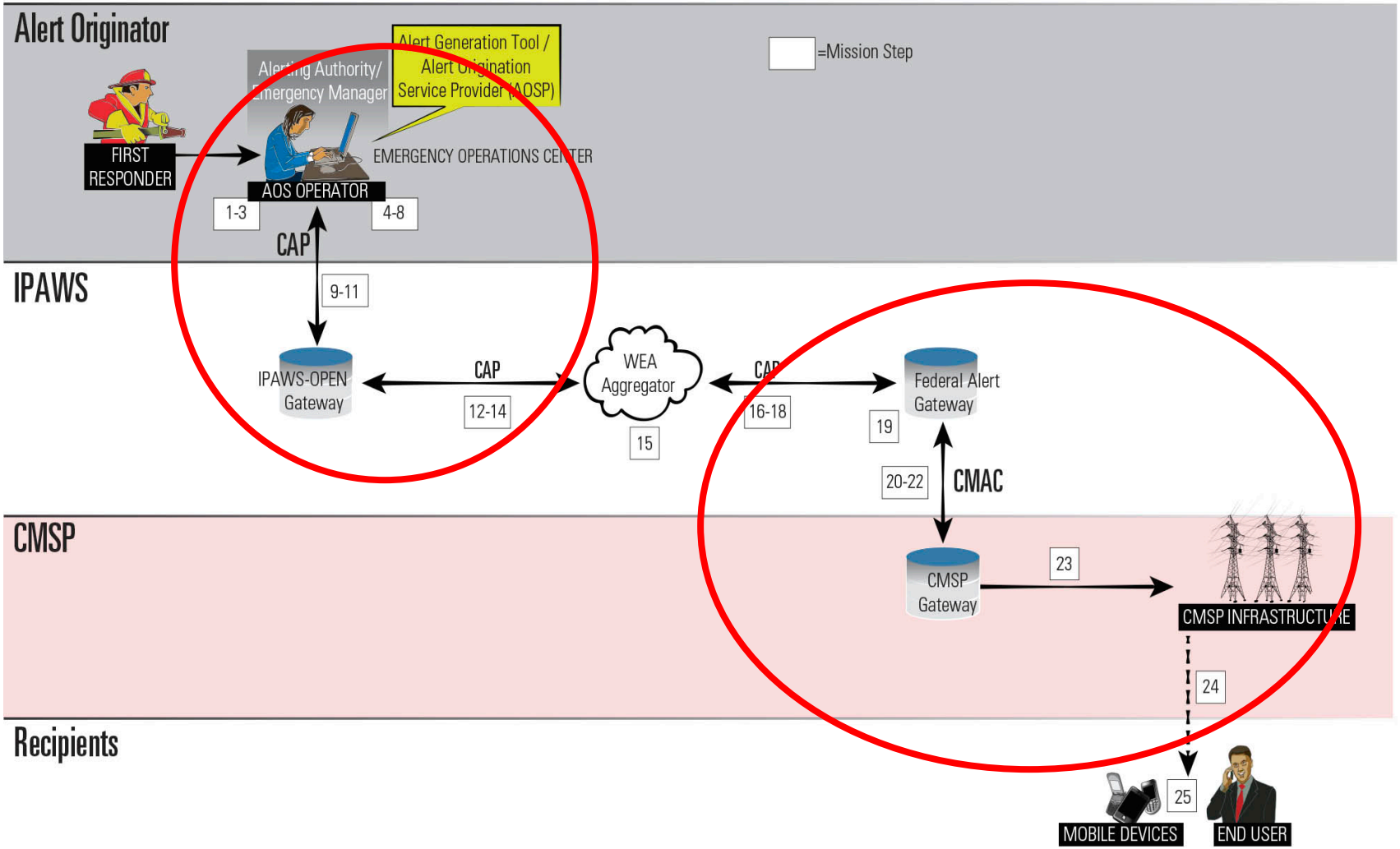
Wireless Emergency Alerting (WEA)

- **Developed a WEA Integration Strategy**
  - Aid Alert Originators (AOs) in adopting and utilizing WEA
- **Developed WEA Best Practices**
  - Develop and publish a collection of WEA Best Practices for AOs
  - Develop and exercise a WEA Trust Model for AOs
- **Developed a WEA Cybersecurity Risk Management (CSRM) strategy for**
  - Alert Originators to assist in their acquisition of wireless capabilities
  - Commercial Mobile Service Providers (CMSPs) to assess cybersecurity risks that affect the WEA service and develop WEA cybersecurity control guidelines for CMSPs



WEA Alerting Pipeline

WEA Mission Thread (System of Systems)



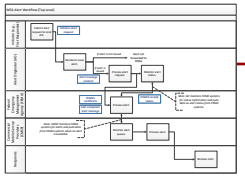
Security Engineering Risk Analysis (SERA)

Outcomes

- Data disclosure (Confidentiality)
- Data modification (Integrity)
- Insertion of false data (Integrity)
- Destruction of data (Availability)
- Interruption of access to data (Availability)

Affects

Workflow / Mission Thread

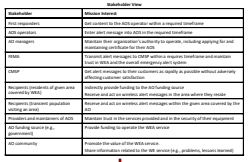


Produces

Adverse Mission Consequences / Losses

Affects

Stakeholder Interests

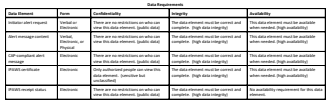


Produces

Adverse Stakeholder Consequences / Losses

Produces

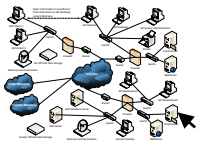
Mission Data



Targets

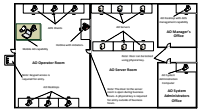
Technology Environment

Technology Infrastructure



Use Cases

Facilities



Entity of Interest

Exploits weaknesses and vulnerabilities

Threat Actor

Research Objectives for Software and Growing Insecurity in SoSE applications

Growing Software Insecurity Objectives

Measures needed to differentiate good from bad software

- Reliable, quick, easy measures of application insecurity
  - Static analysis tools are not readily integrated into an IDE
  - Defect tracking is subjective, inconsistent, and easily skewed
  - No current tools include context
- Fast and low cost measures of compositional security/insecurity
  - Inconsistencies in assumptions among components can create a highly insecure composition from quality parts
  - **Trusted Computer System Evaluation Criteria** (1983-1999), better known as the Orange Book, took too long, cost too much, and did not scale to current demand

Anyone Can Write Software

How To Raise The Next Zuckerberg: 6 Coding Apps For Kids

http://readwrite.com/2013/04/19/how-to-raise-the-next-zuck-6-coding-apps-for-kids/

TYNKER - We Empower KIDS to Become Makers

https://www.tynker.com/

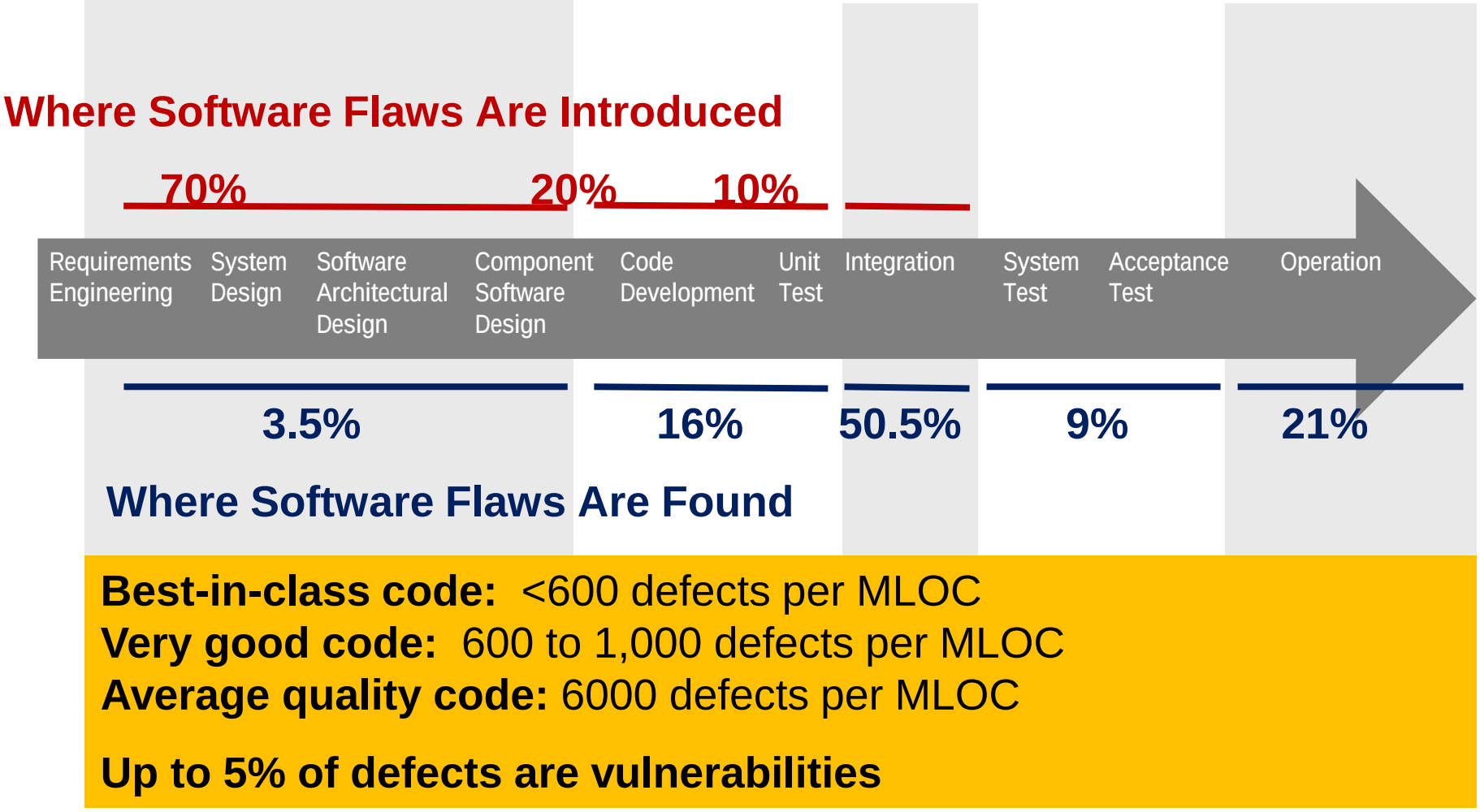
How and Why to Teach Your Kids to Code

http://lifehacker.com/how-and-why-to-teach-your-kids-to-code-510588878

From 1997 to 2012, software industry production grew from $149 billion to $425 billion

From 1990 to 2012, business investments in software grew at more than twice the rate of all fixed business investments; and from 2010 to 2012, software accounted for 12.2 percent of all fixed investment, compared to 3.5 percent for computers and peripherals

Measuring the Growing Defects



Best-in-class code: <600 defects per MLOC
Very good code: 600 to 1,000 defects per MLOC
Average quality code: 6000 defects per MLOC

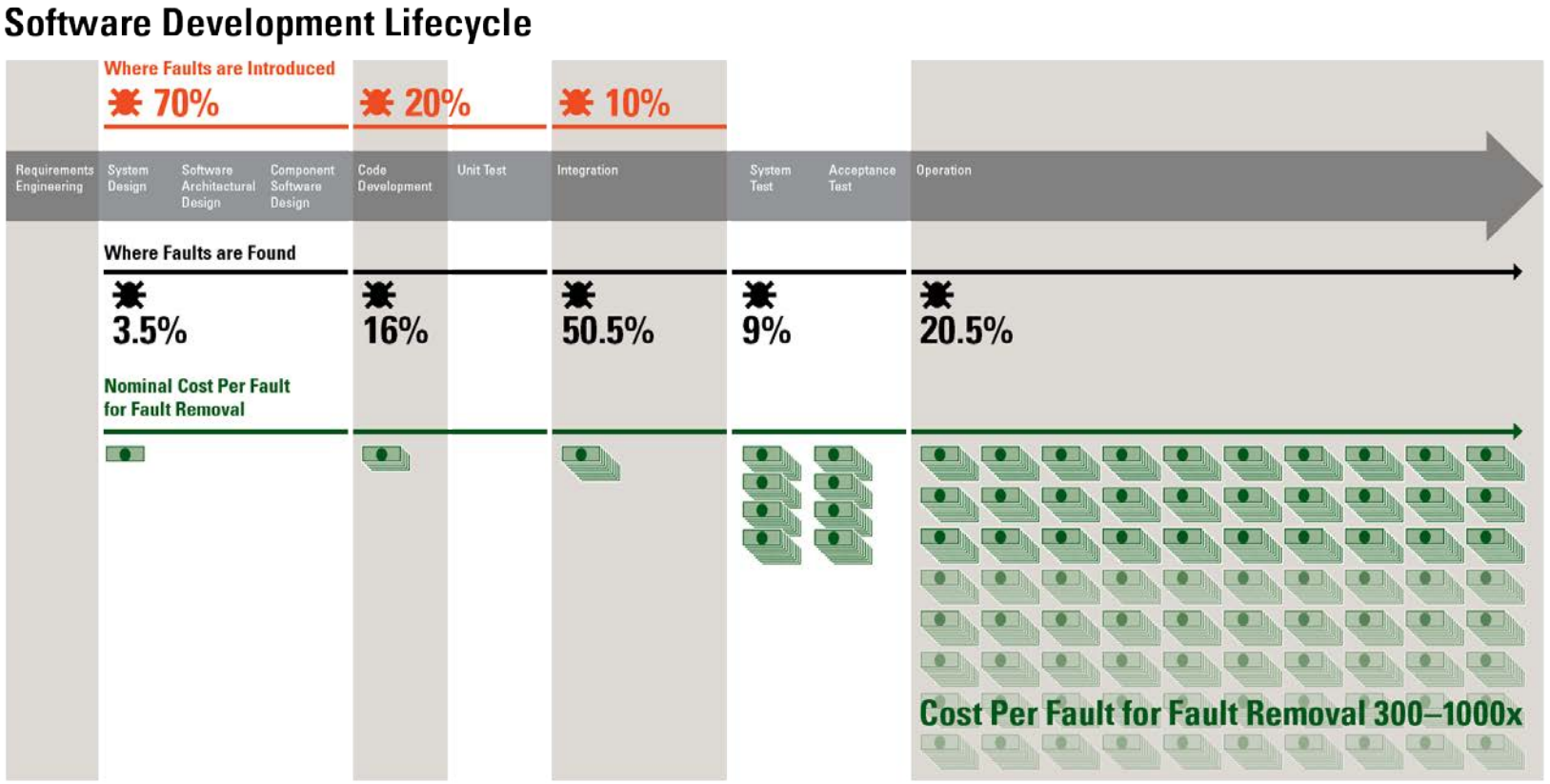
Up to 5% of defects are vulnerabilities

Sources: *Critical Code*; NIST, NASA, INCOSE, and Aircraft Industry Studies

Software Faults: *Introduction, Discovery, and Cost*

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

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



Engineering Software for Security Objectives

Quantitative software measures

- Reliably, quickly, and easily determine software production quality (process, product, use) – many options but no notion of what is most useful
- Measures for predicting/confirming software qualities early in the lifecycle (e.g. security, resilience, survivability) in evaluating “fit for use” and “best buy” choices

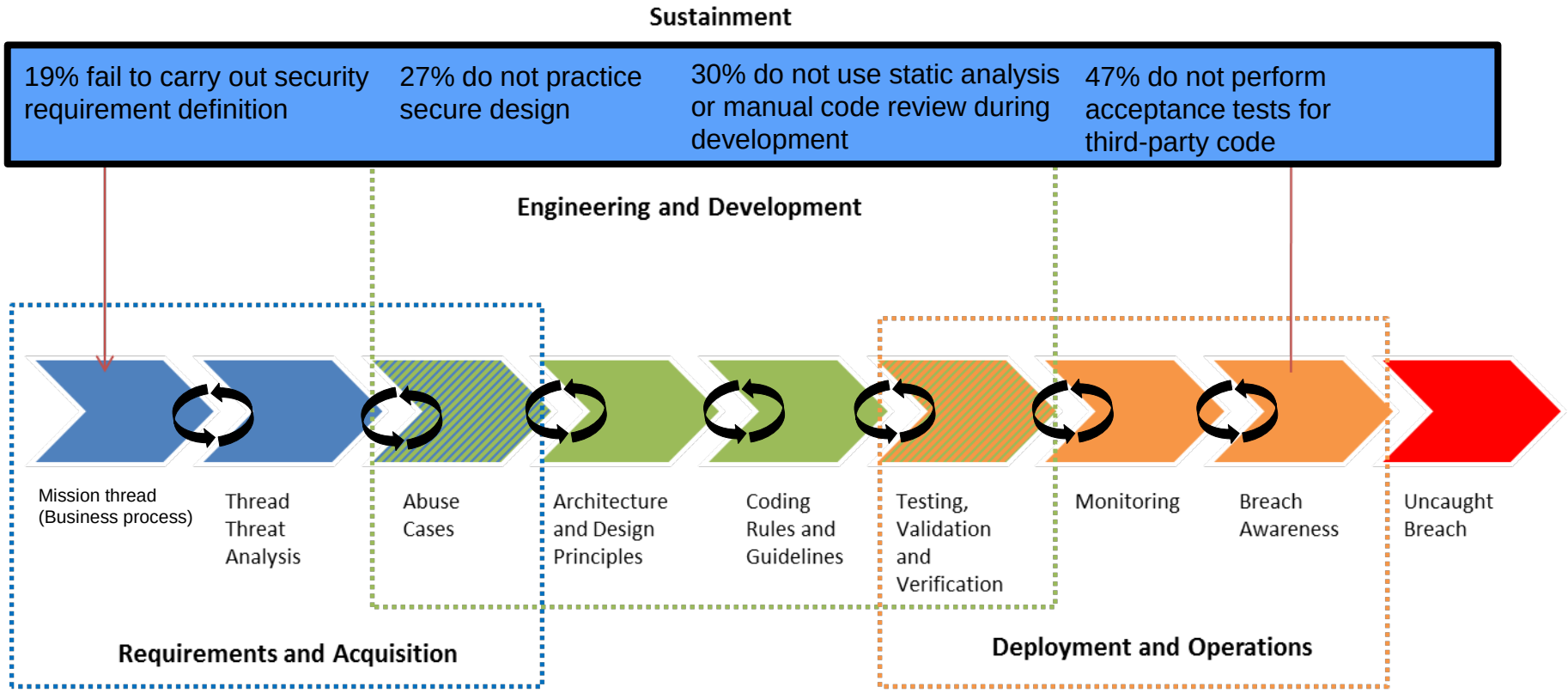
Building code for building software with desired qualities

Software supply chain evaluation mechanisms to differentiate good and poor software suppliers (and their supply chains)

Sample: 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

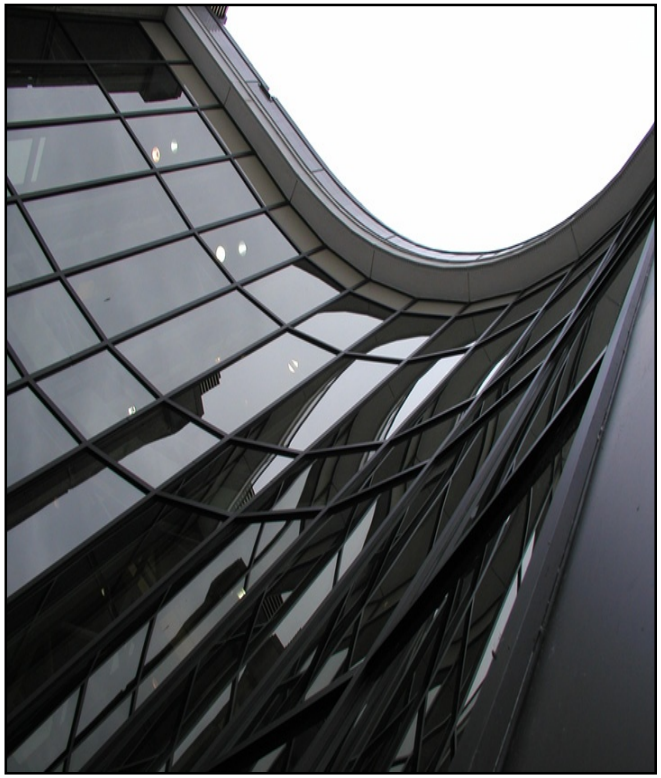
Opportunities for Security Improvement



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

Source: Forrester Consulting, “State of Application Security,” January 2011

Contact Information



Carol Woody, Ph.D.

cwoody@cert.org

Web Resources (SEI)

http://www.sei.cmu.edu/