Challenges of Integrating Software Assurance Engineering Activities into the System Acquisition Life Cycle

Dr. Kenneth E. Nidiffer

28th Annual IEEE Software Technology Conference
National Institute of Standards and Technology
Gaithersburg, MD

25–28 September 2017

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Software Engineering Institute | Carnegie Mellon University

Distribution Statement A: Approved for Public Release; Distribution is Unlimited

Challenges of Integrating Software Assurance Engineering Activities into the System Acquisition Life Cycle

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.

DM17-0605

Software-Enabled Systems Are Today’s Strategic Resource

Dr. Bill Scherlis*

“Software is the building material for modern society”

Manual Labor

Water

Steam

Oil

Software

Increasing Globalization, Productivity, and Complexity

Source: SEI

Context: Increasingly Software Assurance Is a Moving Target

- **Definition:** Software assurance provides the required level of confidence that software functions as intended (and no more) and is free of vulnerabilities, either intentionally or unintentionally designed or inserted in software, throughout the life cycle*
- **Perspective:** The changing and expanding role that software plays in cyberspace means that the development of software-intensive systems must continue to evolve while we pursue software assurance

Source: SEI

*** NDAA 2013, Section 933**

Challenges: Integrating Software Assurance Engineering Activities into the System Acquisition Life Cycle

1. Increasing complexity of software-intensive systems
2. Satisfying unique operational mission and business needs
3. Solving the vulnerability identification chasm
4. Addressing system sustainment
5. Handling the expanding code base
6. Understanding attack patterns, vulnerabilities, and weaknesses
7. Increasing vulnerabilities
8. Designing-in software quality throughout the life cycle
9. Reducing technical debt
10. Working in the infancy of the software engineering discipline

Context: Software Assurance/Cyber Imperative

- Software is a foundation of the DoD’s military power and the building material for modern society
  - *Software assurance is a moving target*
- The Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics (USD(AT&L)) updated DoDI 5000.02 to include a new Enclosure 14 - 2017. The policy states, in part,
  - *Program managers, assisted by supporting organizations to the acquisition community, are responsible for the cybersecurity of their programs, systems, and information…”*
- Direct link between cybersecurity engineering and systems and software assurance engineering*

**Cyber Security Engineering: A Practical Approach for Systems and Software Assurance*, Carol Woody and Nancy Mead, 2017

Context: Dynamics of Software

- Software is ubiquitous and growing in importance
- Codebases are increasing
- Vulnerabilities (defects, flaws) are increasing
- Software represents increasingly more system functionality and cost

- Research is needed to address emerging software challenges
- Software-reliant systems are becoming more complex and intertwined
- There is national and global dependence on software
- We need to improve the management of software-intensive systems
- Software assurance is increasingly important, and achieving it is a moving target

Context: The Fabric of Computing Is Changing and Achievement of Software Assurance Is More Challenging

Source: SEI

Networks are becoming software defined

- Generic network nodes can adapt function to usage and demand

Field-Programmable Gate Arrays (FPGAs) are proliferating

- Now even the processor's function is determined by software and malleable after fielding

Emergence of Artificial Intelligence and Machine Learning

- Tasks change from direct programming to data curation and feature discovery

Context: The DoD Should Expect Cyber Attacks to Be Part of All Conflicts in the Future*

Hacktivist

Criminal

Espionage

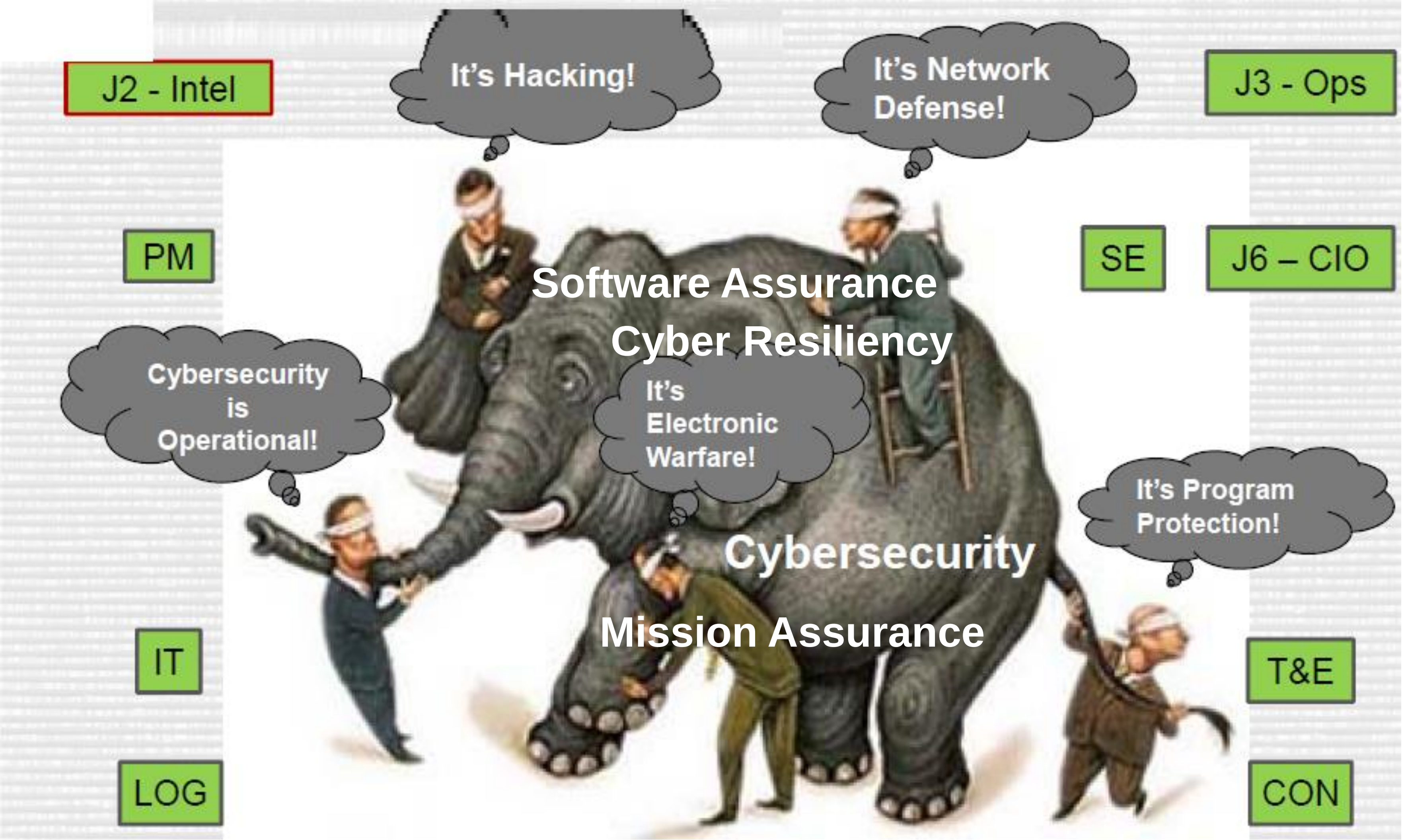
Terrorism

State-Sponsored Disruptions/War

> “The DoD should not expect competitors to play by our version of the rules”*

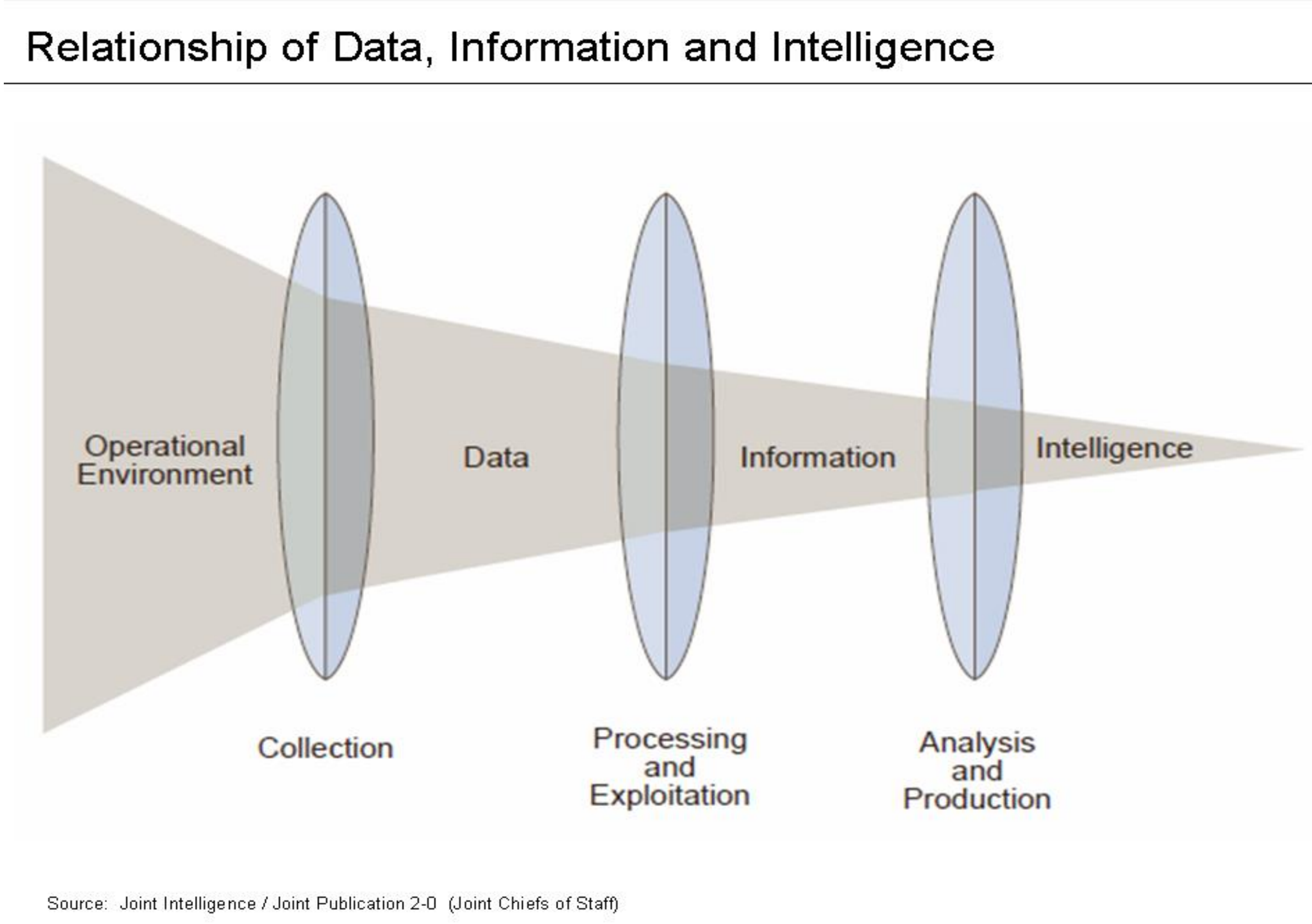
*Defense Science Board (DSB) Report, Jan 2013

Context: DoD Stakeholders and Different Perspectives on Software Assurance



Source: DAU

Context: Effective Decision Making Increasingly Depends on Software Assurance

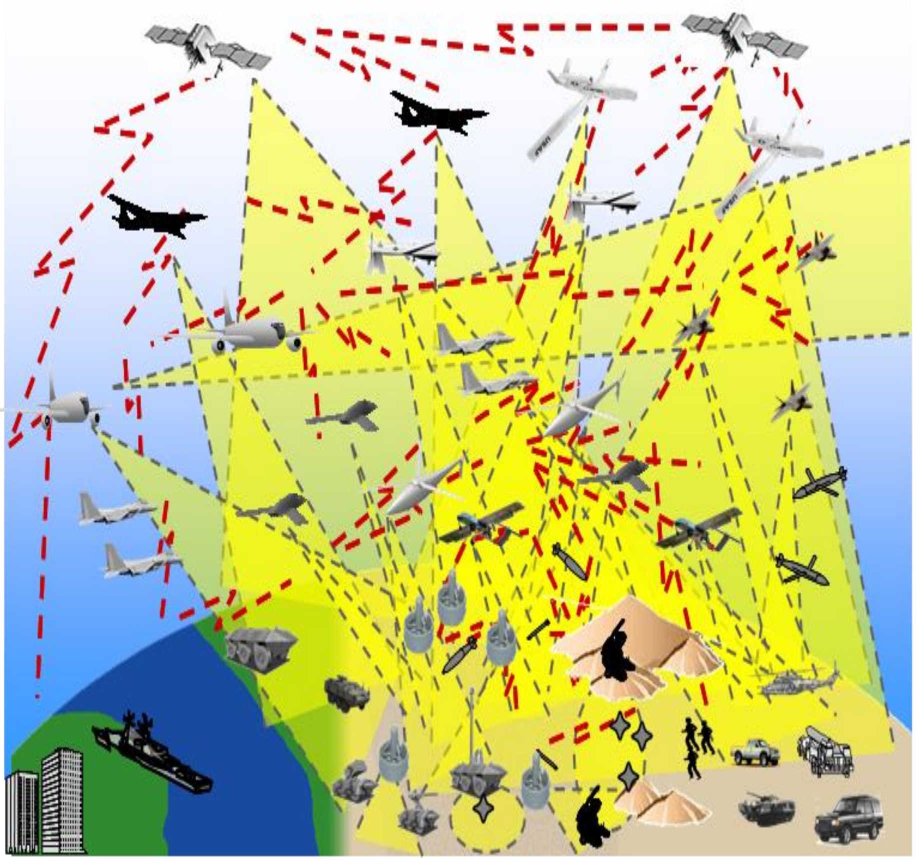

Relationship of Data, Information and Intelligence

Source: Joint Intelligence / Joint Publication 2-0 (Joint Chiefs of Staff)

Decision Maker

Context: Enduring Questions That Drive Hard Choices About Software Assurance

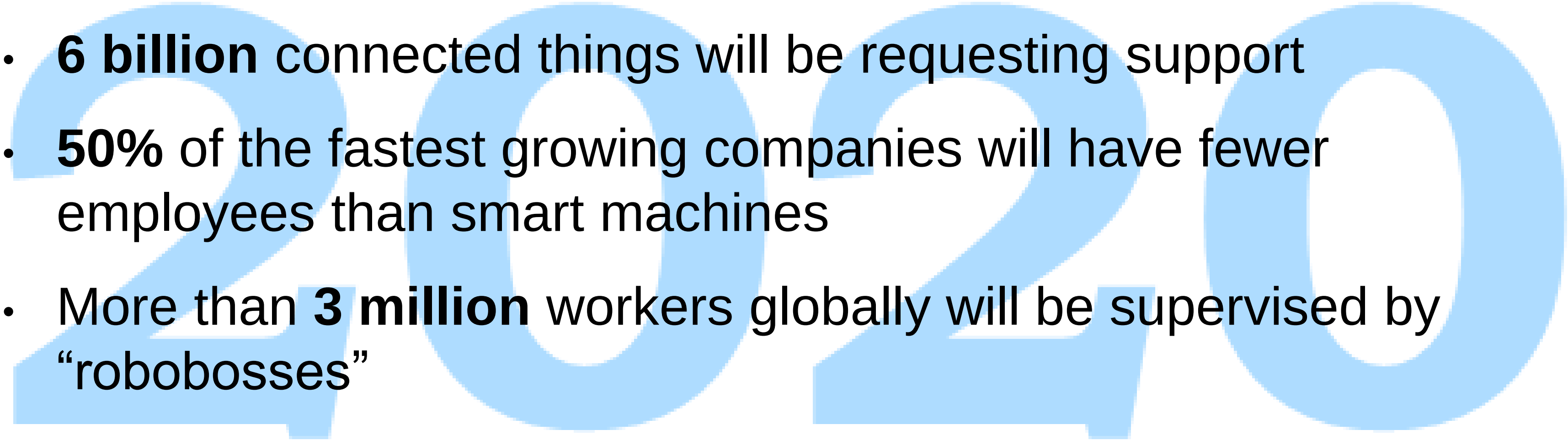
- How much is “enough software assurance”?
- How much does “enough” cost?
- Is “enough” affordable?
- How does one decide?
- How does one evaluate the “goodness” of the decision?



Technical advancements in software assurance achievement, with operational participation, are needed

Source: SEI

Context: Future – Autonomous Systems Domain*



- Algorithmically driven agents will work in **5%** of economic transactions
- **20%** of all business content will be authored by machines
- **6 billion** connected things will be requesting support
- **50%** of the fastest growing companies will have fewer employees than smart machines
- More than **3 million** workers globally will be supervised by “robobosses”

DoD is increasingly employing autonomous capabilities across a diverse number of systems

Source: DSB Study – June 2016

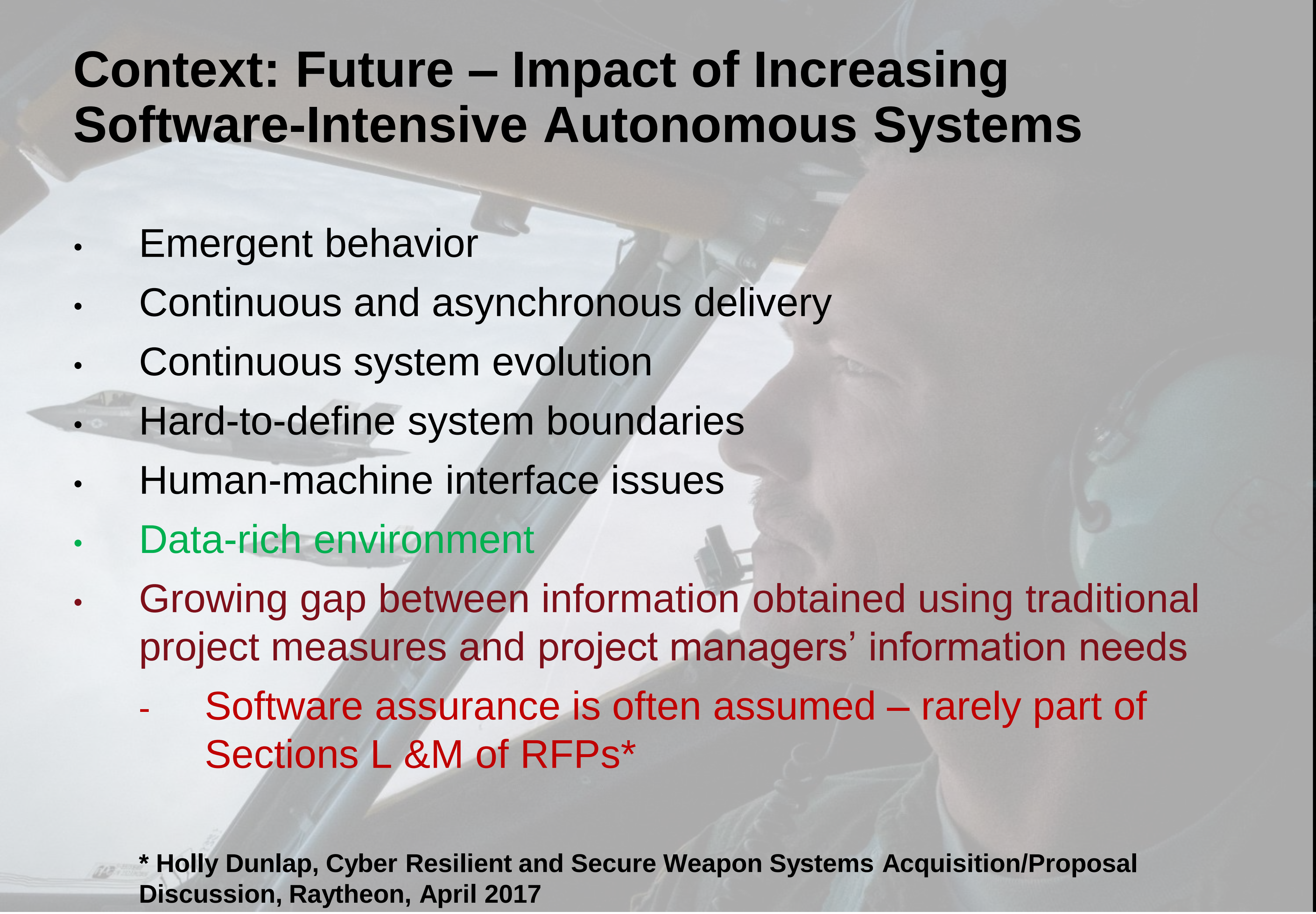
Context: Future – Autonomous Systems in Use Today and in the Future Are the Result of Decades of R&D

R&D areas include

- digitization of sensors
- adaptive algorithms
- natural user interfaces
- machine learning
- machine vision
- data analytics

Source: SEI

Context: Future – Impact of Increasing Software-Intensive Autonomous Systems

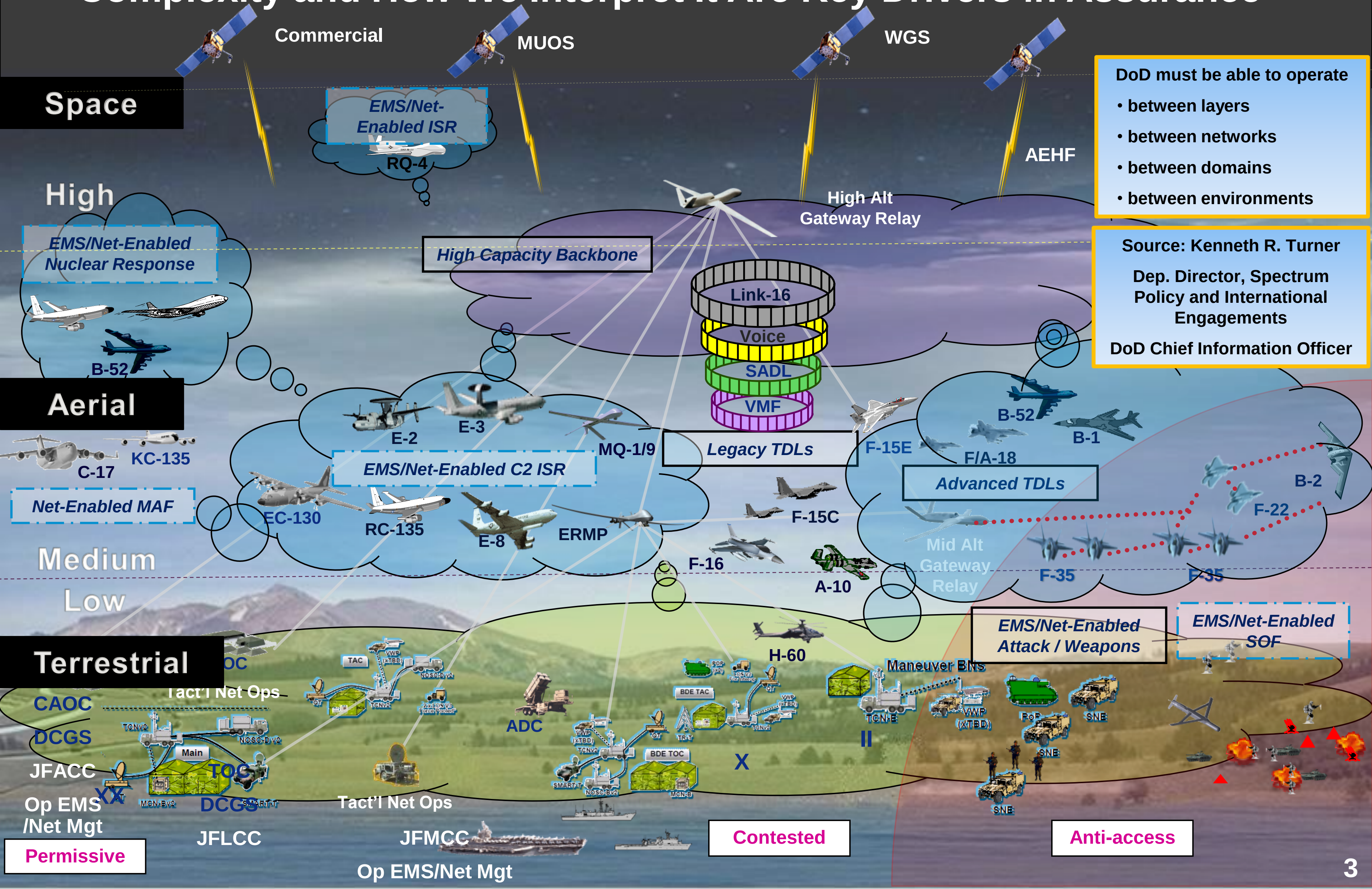


- Emergent behavior
- Continuous and asynchronous delivery
- Continuous system evolution
- Hard-to-define system boundaries
- Human-machine interface issues
- Data-rich environment
- Growing gap between information obtained using traditional project measures and project managers' information needs
  - Software assurance is often assumed – rarely part of Sections L &M of RFPs*

*** Holly Dunlap, Cyber Resilient and Secure Weapon Systems Acquisition/Proposal Discussion, Raytheon, April 2017**

Increasing Complexity of Cybersecurity Systems

Complexity and How We Interpret It Are Key Drivers in Assurance



DoD must be able to operate

- between layers
- between networks
- between domains
- between environments

Source: Kenneth R. Turner

Dep. Director, Spectrum Policy and International Engagements

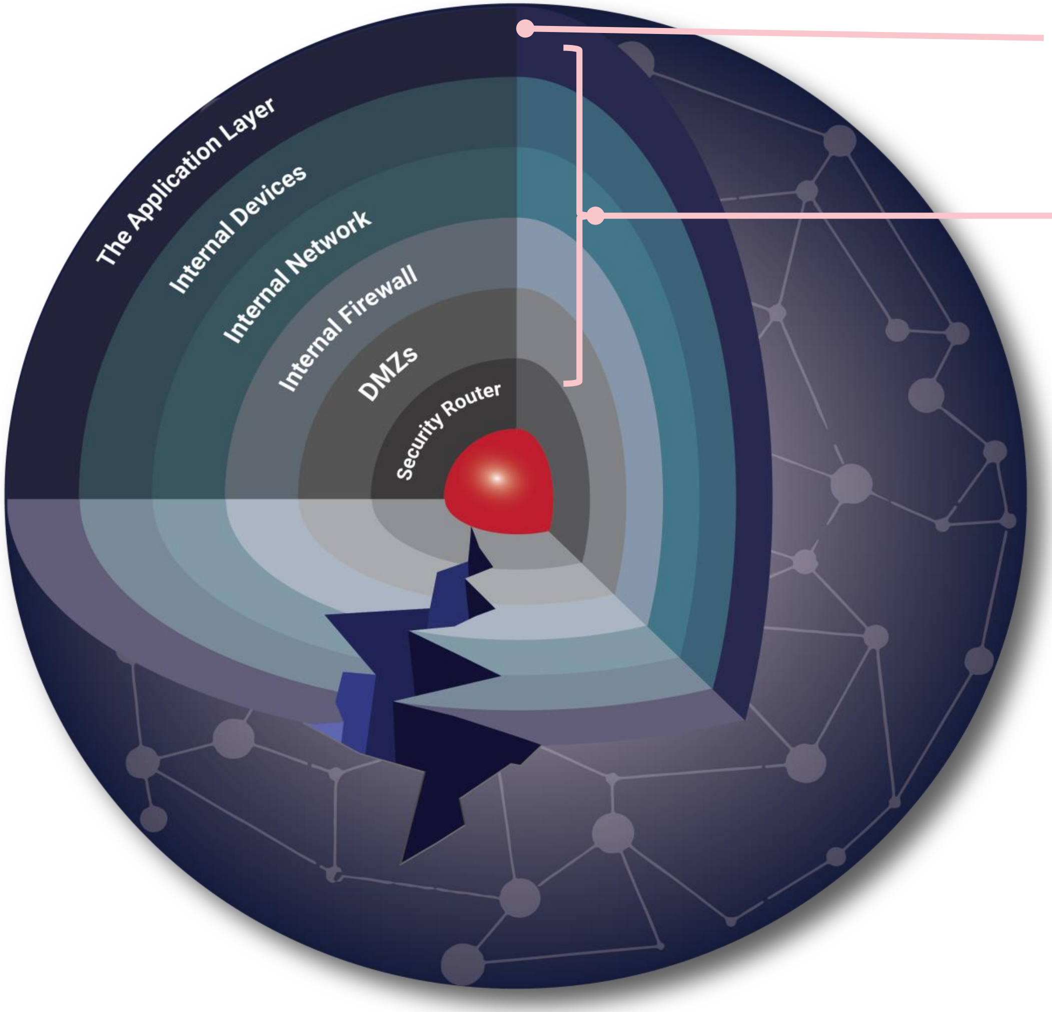
DoD Chief Information Officer

Satisfying Unique Operational Mission and Business Needs as Commercial Products Are Integrated into Military Systems

Source: SEI

Solving the Vulnerability Identification Chasm

First line of defense in software assurance is the application (software) layer



84% of breaches exploit vulnerabilities in the application[1]

Yet funding for IT defense vs. software assurance is 23 to 1[2]

[1] Clark, Tim, “Most Cyber Attacks Occur from This Common Vulnerability,” *Forbes*, 03-10-2015

[2] Feiman, Joseph, “Maverick Research: Stop Protecting Your Apps; It’s Time for Apps to Protect Themselves,” *Gartner*, 09-25-2014. G00269825

Addressing System Sustainment

Software assurance development and sustainment activities need to be integrated across the entire system life cycle*

Source: SEI

Break point where software is handed off for sustainment is increasing blurred

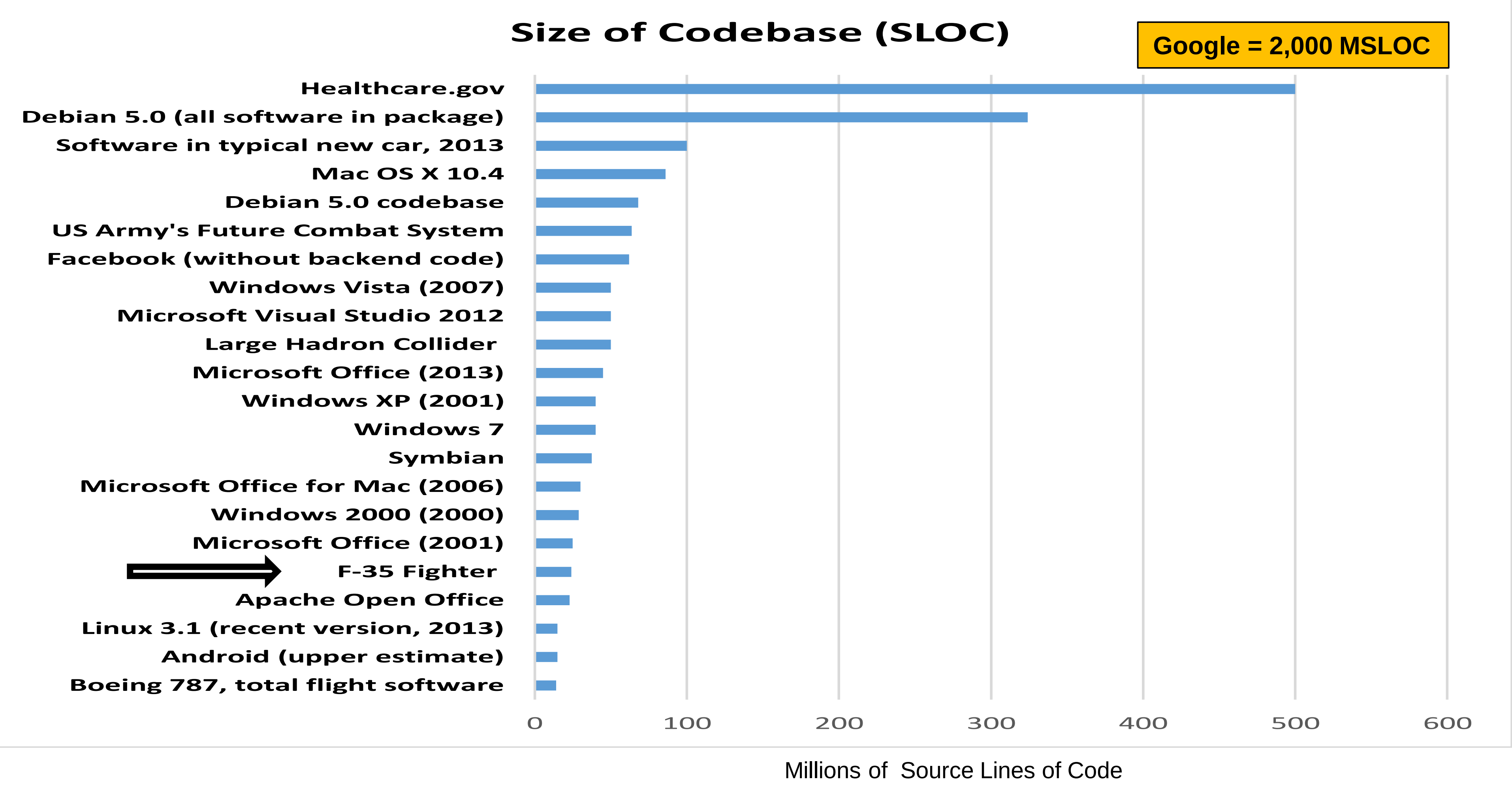
Involves coordinating processes, procedures, people, and information

Challenges include

- rising costs
- recertification/retesting
- dynamic operating environments
- legacy environments
- Life-cycle SwA activities and measures

Handling the Expanding Code Base

Software is dramatically expanding with limited natural governance

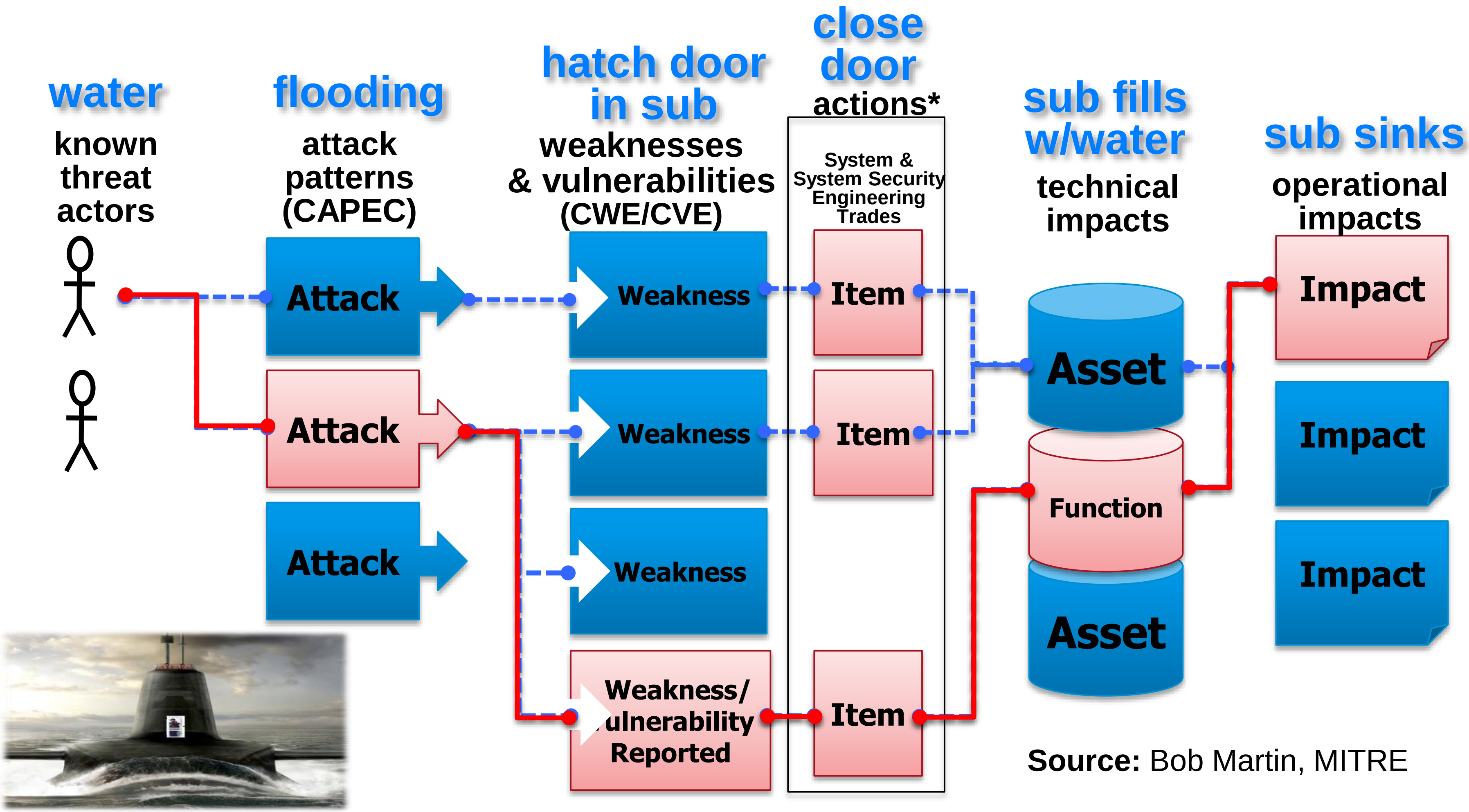


Google = 2,000 MSLOC

Source: David McCandless, “Information is Beautiful,” 21 September 2016 web retrieval

Understanding Attack Patterns, Vulnerabilities, and Weaknesses

Defining software assurance attributes to satisfy information needs



water
known threat actors

flooding
attack patterns (CAPEC)

hatch door in sub
weaknesses & vulnerabilities (CWE/CVE)

close door actions*
System & System Security Engineering Trades

sub fills w/water
technical impacts

sub sinks
operational impacts

Attack
Attack
Attack

Weakness
Weakness
Weakness
Weakness/ Vulnerability Reported

Item
Item
Item

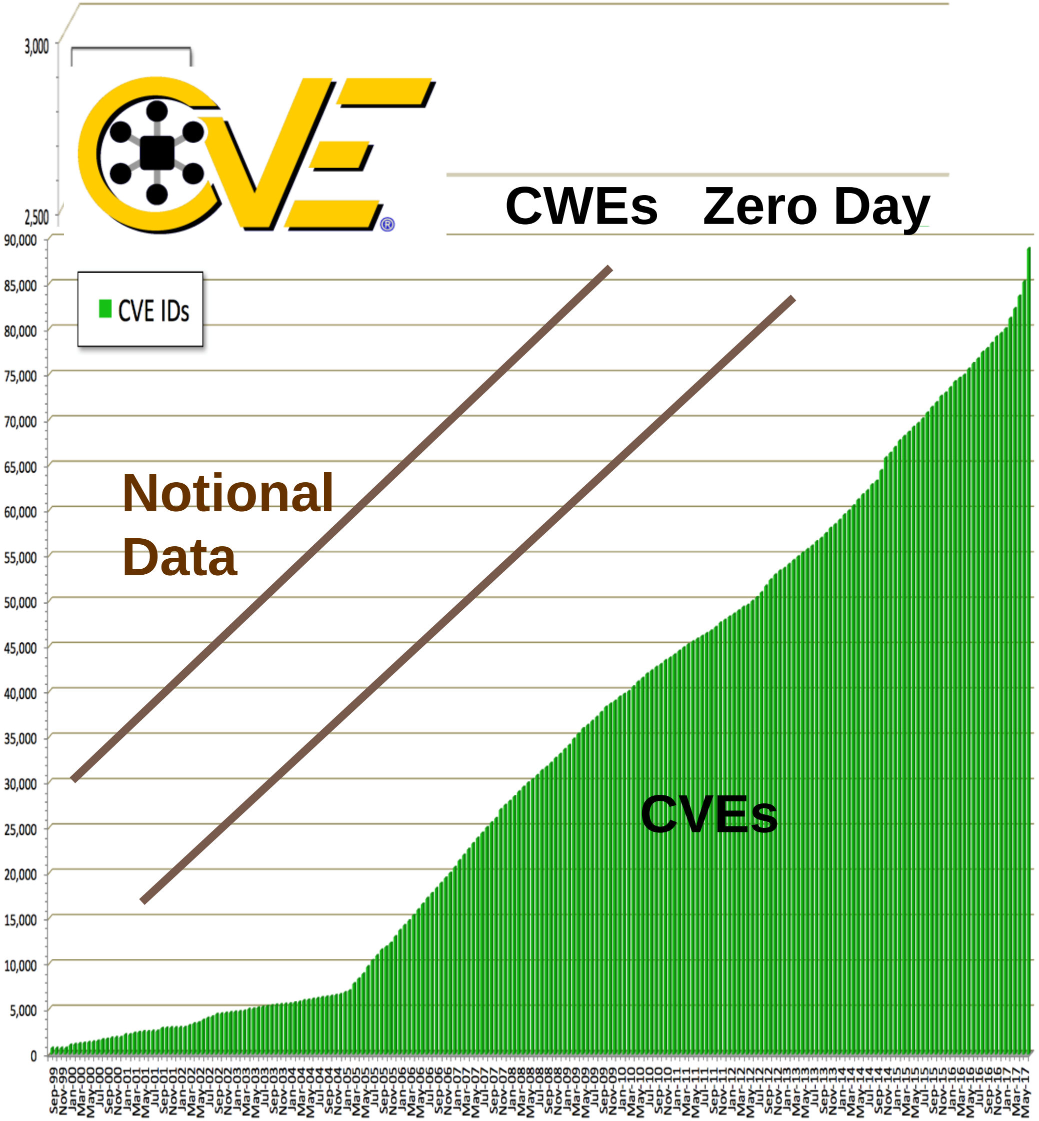
Asset
Function
Asset

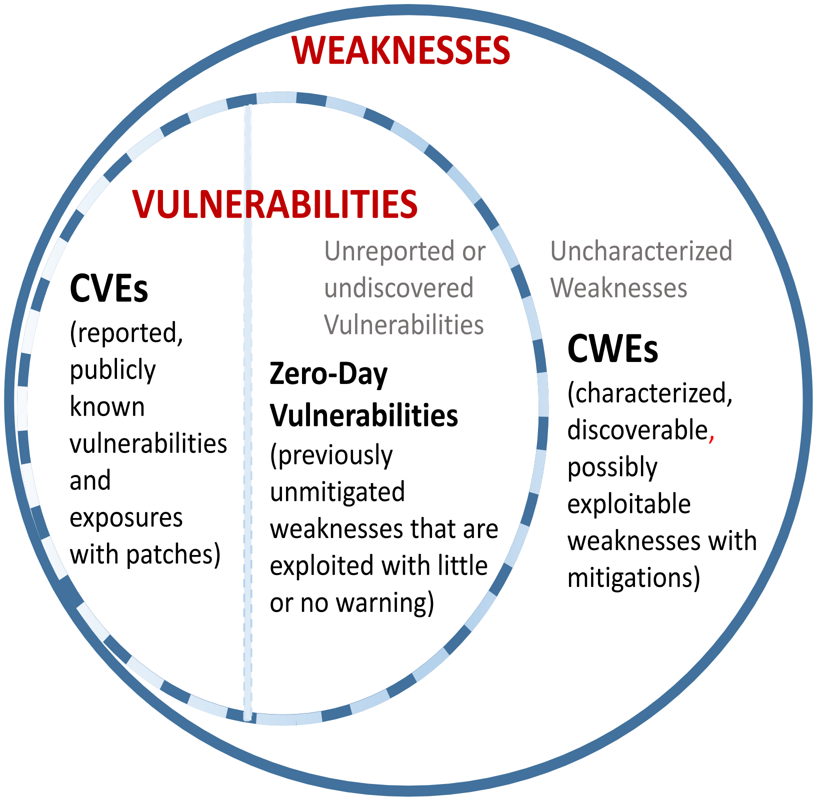
Impact
Impact
Impact

Source: Bob Martin, MITRE

- **"Actions"** include architecture choices; design choices; added security functions, activities, and processes; physical decomposition choices; static and dynamic code assessments; design reviews; dynamic testing; and pen testing.
- **Vulnerability** is the intersection of three elements: a system susceptibility or flaw, attacker access to the flaw, and attacker capability to exploit the flaw.

Increasing Vulnerabilities: CVE 1999 to 2017: Reported Common Vulnerabilities and Exposures (CVE)




WEAKNESSES
VULNERABILITIES
Unreported or undiscovered Vulnerabilities
Uncharacterized Weaknesses

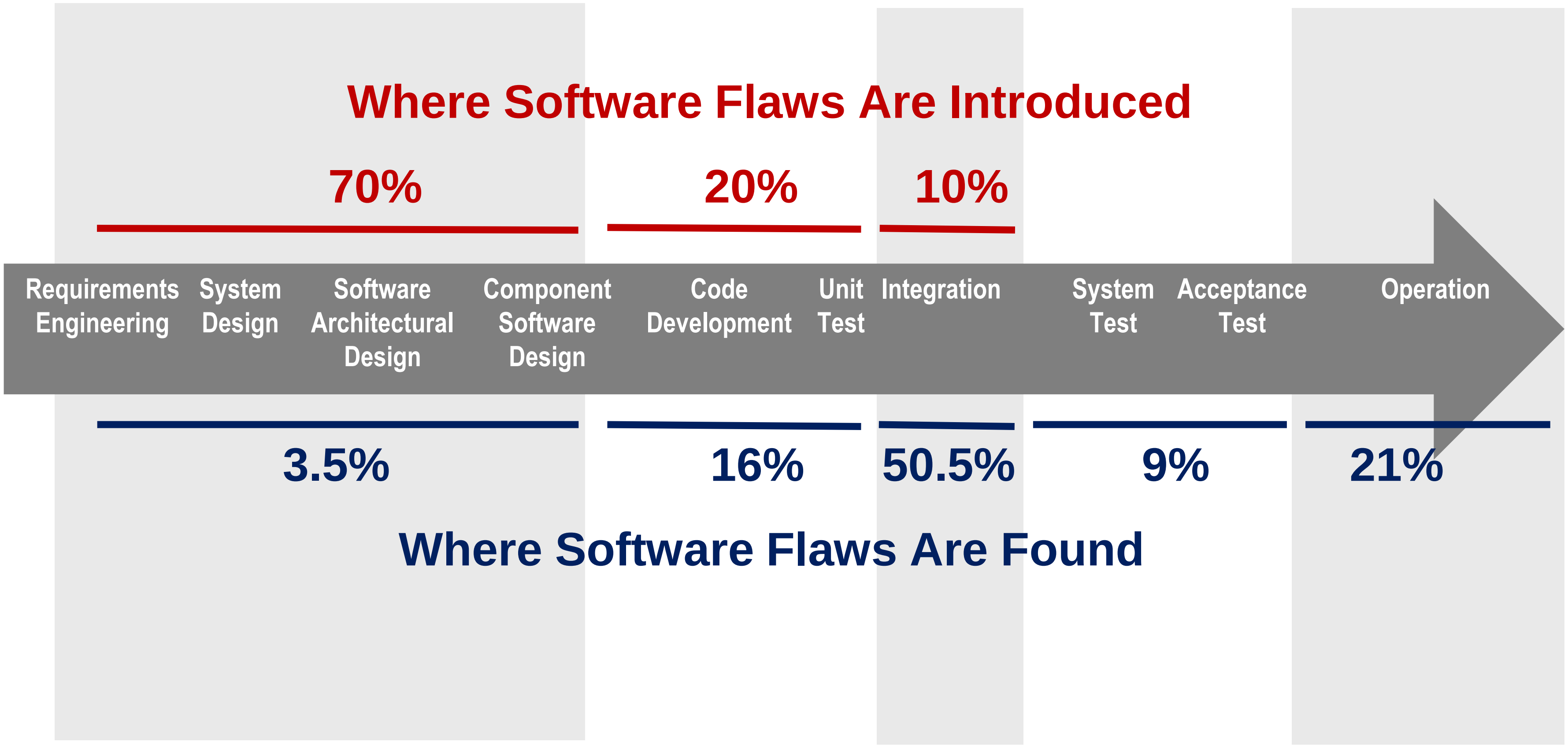
CVEs (reported, publicly known vulnerabilities and exposures with patches)

Zero-Day Vulnerabilities (previously unmitigated weaknesses that are exploited with little or no warning)

CWEs (characterized, discoverable, possibly exploitable weaknesses with mitigations)

Source: Dr. Robert A. Martin, MITRE Corporation, May 2017

Designing-in Software Assurance Throughout the System Life Cycle



Where Software Flaws Are Introduced

70% 20% 10%

Requirements Engineering | System Design | Software Architectural Design | Component Software Design | Code Development | Unit Test | Integration | System Test | Acceptance Test | Operation

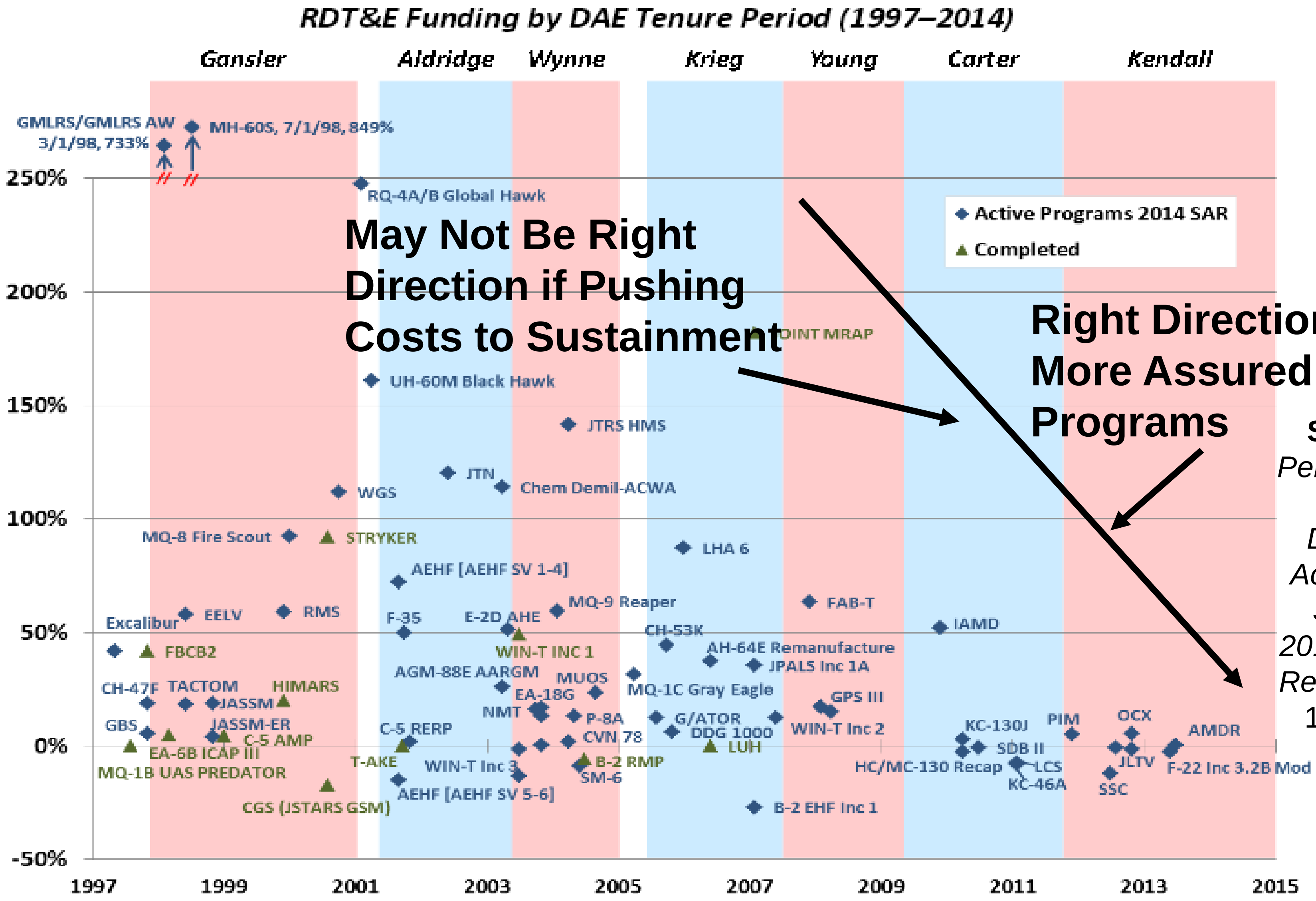
3.5% 16% 50.5% 9% 21%

Where Software Flaws Are Found

Special emphasis needed up-front in the system life cycle

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

Reducing Technical Debt



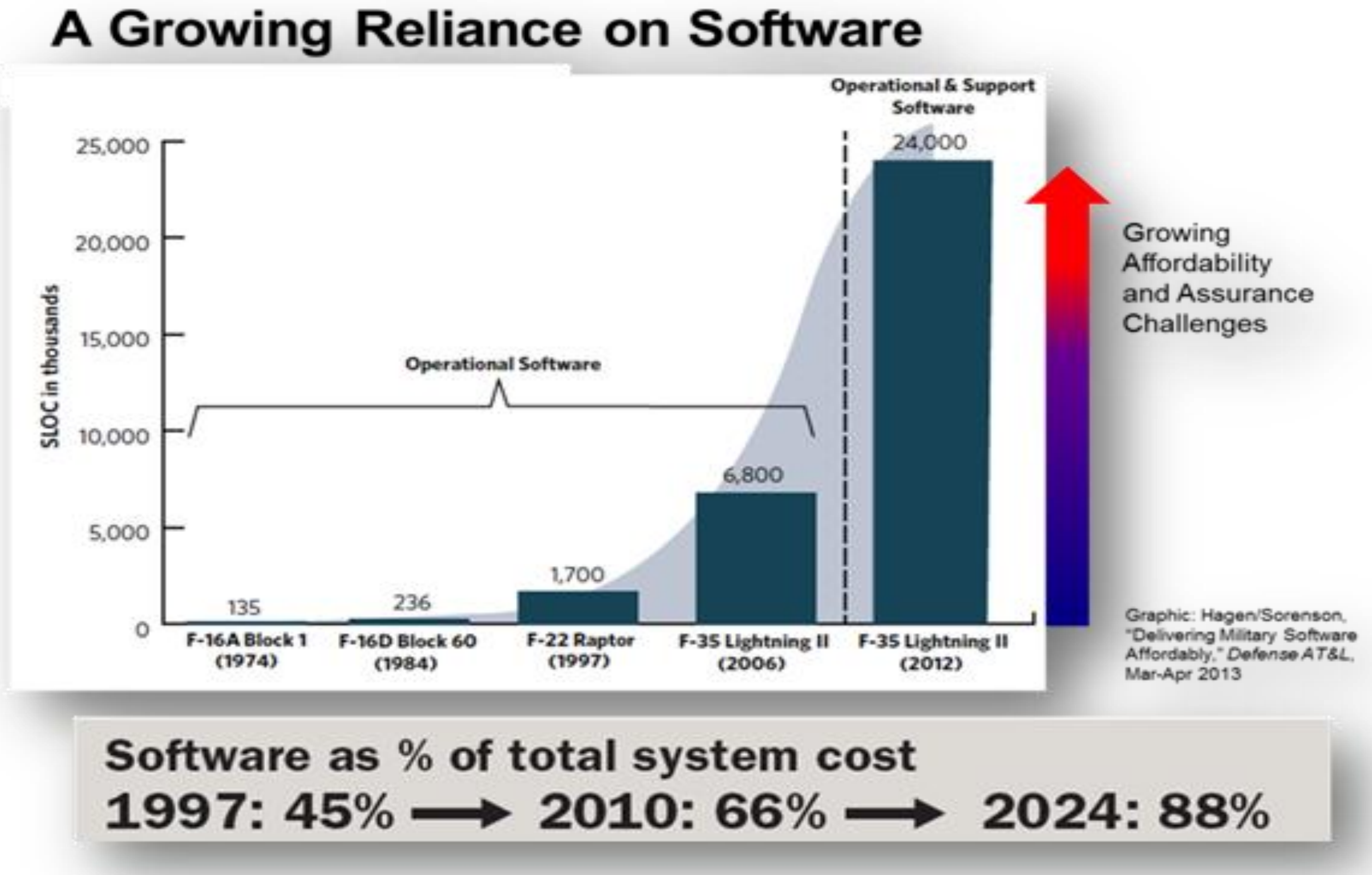
May Not Be Right Direction if Pushing Costs to Sustainment

Right Direction if More Assured Programs

Source: *Performance of the Defense Acquisition System 2015 Annual Report*, Sep. 16, 2015

Distribution Statement A: Approved for Public Release; Distribution is Unlimited

Reducing Technical Debt Over the System Life Cycle



A Growing Reliance on Software

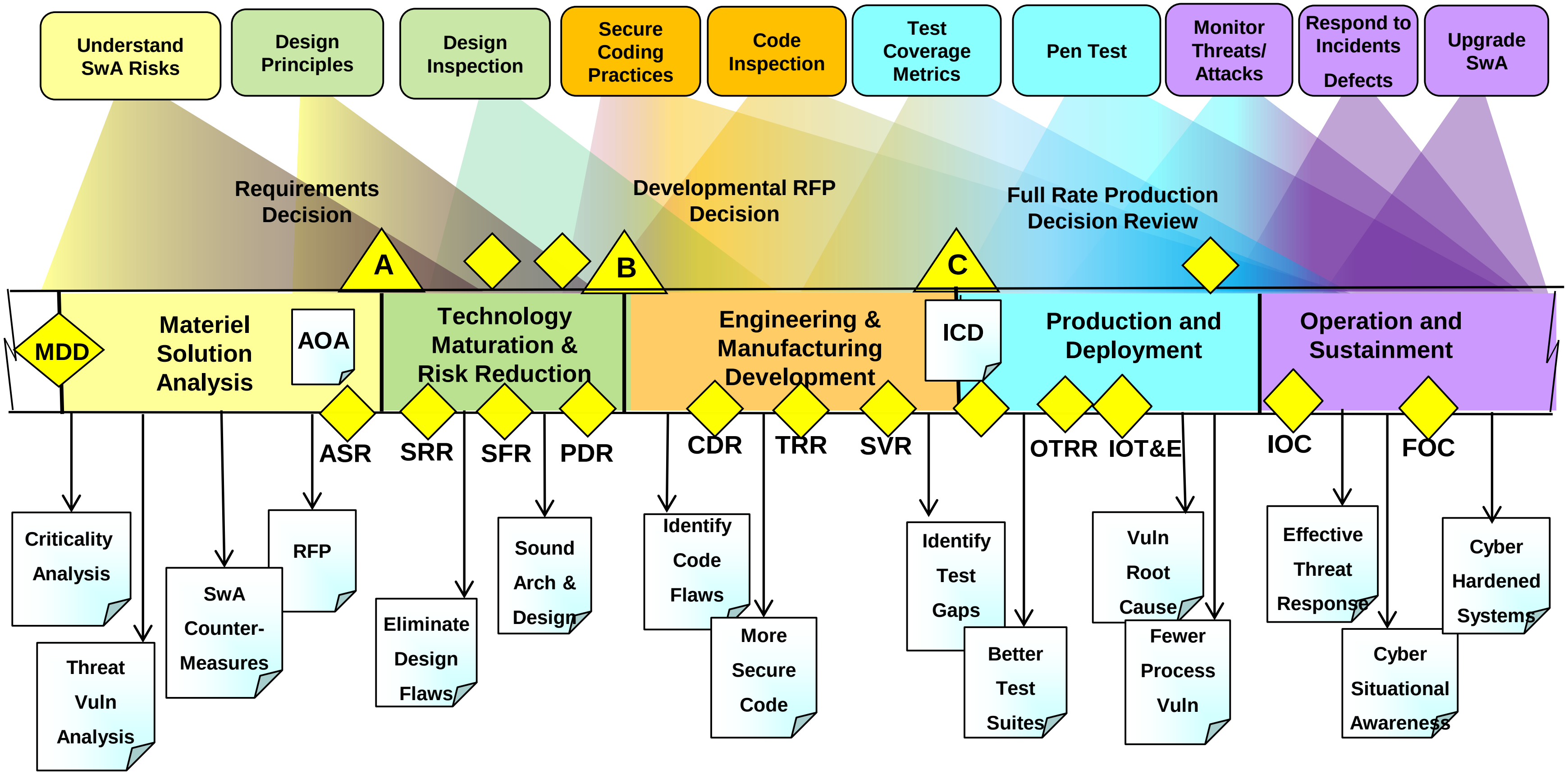
Graphic: Hagen/Sorenson, "Delivering Military Software Affordably," *Defense AT&L*, Mar-Apr 2013

Software as % of total system cost

1997: 45% → 2010: 66% → 2024: 88%

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.

Reducing Technical Debt: Engineering-in Software Assurance Activities Across the Life Cycle



Working in the Infancy of the Software Engineering Discipline

Improving the workforce by developing software core competencies and a DoD career field in software engineering

	Physical Science	Bioscience	Computer/Software/Cyber Science
Origins/History	Begun in antiquity	Begun in antiquity	Mid-20th century
Enduring Laws	Laws are foundational to furthering exploration in the science	Laws are foundational to furthering exploration in the science	Only mathematical laws have proven foundational to computation
Framework of Scientific Study	Four main areas: astronomy, physics, chemistry, and earth sciences	Science of dealing with health maintenance and disease prevention and treatment	• Several areas of study: computer science, software/systems engineering, IT, HCI, social dynamics, AI • All nodes are attached to and rely on a netted system
R&D and Launch Cycle	10–20 years	10–20 years	Significantly compressed; solution time to market must happen very quickly

HCI: human–computer interaction; AI: artificial intelligence

Source: SEI

Infancy of Software Engineering Discipline: Human-Machine Teaming

In the real world, autonomy is usually granted within some context—explicit or implicit

- parents and children
- soldiers, sailors, marines, and airmen

How do we do this for machines?

- Explicit may be easy, but implicit is hard for machines
- Commander's intent
- Mission orders

Related to need for explainability and predictability

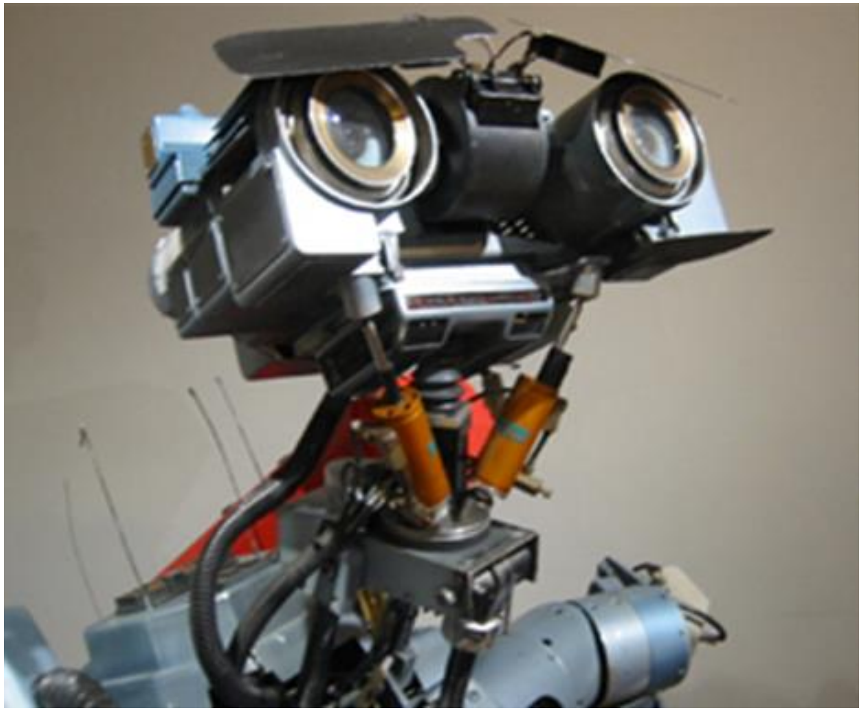
Source: SEI

So Where Does This Lead Us?

- A more robust software assurance approach will be needed…
- Decision makers will need insight and understanding about how to achieve software assurance
- As software-dominated system projects become larger in scope/complexity, capitalizing on opportunities for making better decisions will become more important
  - Critical to shift from asking “what happened?” which is a question of information based on sparse data
  - To seeking insight by asking “what happened, why, how do we solve the problem, and can we evaluate that it has been solved?”
- Enabling an engineering-based approach that seeks to design-in software assurance is becoming more important
- DoD workforce needs a software engineering career field that includes software assurance core competencies

Final Thought: Advanced Software Engineering with Operational Participation

Will determine if we create C-3PO and Johnny 5 . . .

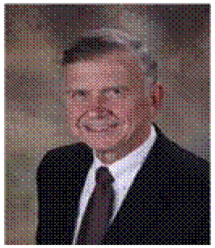


Source: SEI

…or the Borg

Source: SEI

Contact Information



Dr. Kenneth E. Nidiffer, Director of Strategic Plans for Government Programs

Software Engineering Institute
Carnegie Mellon University
Office: + 1 703-247-1387
Fax: + 1 703-908-9235
Email: Nidiffer@sei.cmu.edu